Redbooks

ibm.com/redbooks

IBM Spectrum Scale (formerly GPFS)

Dino Quintero

Luis Bolinches

Puneet Chaudhary

Willard Davis

Steve Duersch

Carlos Henrique Fachim

Andrei Socoliuc

Olaf Weiser

Cloud

Power Systems

IBM

Redbooks

International Technical Support Organization

IBM Spectrum Scale (formerly GPFS)

May 2015

SG24-8254-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (May 2015)

This edition applies to IBM Spectrum Scale (formerly GPFS) v4.1.0.4 (TL1), SLES 11 SP2, SLES11 SP3, RHEL ppc64 6.5, AIX 7.1 TL3, Tivoli Storage Manager 7.1 Server, Client and Storage Agent (LAN-free), Tivoli Storage Manager for Space Management 7.1.1, and TSM for Space Management (HSM) 7.1.1.

© Copyright International Business Machines Corporation 2015. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices ix
Trademarks x

IBM Redbooks promotions xi

Preface xiii
Authors xiii
Now you can become a published author, too! xiv
Comments welcome xv
Stay connected to IBM Redbooks xv

Chapter 1. Introduction 1
1.1 Overview 2
1.2 IBM Spectrum Scale major components and terminology 3
1.2.1 IBM Spectrum Scale cluster topologies 5
1.3 IBM Spectrum Scale new features and enhancements 5
1.3.1 Enhanced security 6
1.3.2 Performance improvements 6
1.3.3 New usability features 7
1.3.4 Reliability, availability, and serviceability 10
1.4 IBM Spectrum Scale competitive strategy 10
1.5 IBM Spectrum Scale licensing 12
1.6 IBM Spectrum Scale on cross-platform environments 15
1.6.1 IBM Spectrum Scale for AIX 15
1.6.2 IBM Spectrum Scale for Windows operating systems 16
1.6.3 IBM Spectrum Scale for Linux 16
1.6.4 IBM Spectrum Scale for Linux on System z 17
1.6.5 IBM Spectrum Scale Server solution 17
1.7 IBM Spectrum Scale and virtualization 18
1.7.1 Virtualization on IBM Power Systems 19
1.7.2 Virtualization on x86 Linux operating systems 26
1.7.3 Virtualization on Windows operating systems 27
1.8 Contact information 28

Chapter 2. Infrastructure planning and considerations 29
2.1 IBM Spectrum Scale cluster topologies 30
2.1.1 Network-based Spectrum Scale client 30
2.1.2 Direct attached storage 30
2.1.3 Mixed NSD access: server and clients 31
2.1.4 IBM Spectrum Scale File Placement Optimizer 32
2.2 Network design 33
2.2.1 Ethernet adapters options 33
2.2.2 NIC teaming configurations 34
2.2.3 InfiniBand networks 36
2.2.4 RDMA over Converged Enhanced Ethernet 39
2.2.5 IBM data center networking products 45
2.3 Storage design 45
2.3.1 Host bus adapter 46
2.3.2 Multipath driver 47

© Copyright IBM Corp. 2015. All rights reserved.

2.3.3 Storage subsystem considerations 50
2.3.4 Tape library 54
2.4 IBM Spectrum Scale supported platforms 56
2.4.1 IBM Spectrum Scale on AIX 57
2.4.2 IBM Spectrum Scale for Linux (x86 and POWER) 57
2.4.3 IBM Spectrum Scale for Linux on System z 59
2.4.4 IBM Spectrum Scale on Windows operating systems 61
2.4.5 Summary of IBM Spectrum Scale support functions on operating systems 64
2.4.6 GPT partition table 66
2.4.7 SCSI-3 Persistent Reservation 66
2.5 Security considerations for IBM Spectrum Scale clusters 69
2.5.1 Remote shell with ssh 70
2.5.2 Remote cluster: Subnets, firewall rules, and TCP port numbers 70
2.5.3 SELinux configuration with IBM Spectrum Scale 71
2.6 IBM Spectrum Scale configuration planning 72
2.6.1 High availability 72
2.6.2 Network Shared Disk creation considerations 78
2.6.3 Planning the IBM Spectrum Scale file system 85
2.6.4 Planning for IBM Spectrum Scale FPO 93

Chapter 3. Scenarios 97
3.1 IBM Spectrum Scale advantages over Network File System 98
3.2 IBM Spectrum Scale in active-passive and mutual takeover clusters 98
3.3 IBM Spectrum Scale in active-active clusters 99
3.4 Two-node Linux IBM Spectrum Scale cluster 99
3.4.1 Installing IBM Spectrum Scale 99
3.4.2 Configure auxiliary tools 100
3.4.3 Building the IBM Spectrum Scale portability layer on Linux nodes 102
3.4.4 Create IBM Spectrum Scale cluster 104
3.4.5 Create NSD disks 105
3.4.6 Starting the IBM Spectrum Scale cluster 107
3.4.7 Quorum configuration 107
3.4.8 Start IBM Spectrum Scale at boot 109
3.4.9 Create the IBM Spectrum Scale file system 109
3.5 Cluster NFS 110
3.5.1 NFS setup 110
3.5.2 Configuring cNFS 111
3.6 Windows operating system-only cluster 112
3.6.1 Configuring Windows operating system 113
3.6.2 Static IP address 113
3.6.3 Active Directory domain 114
3.6.4 UAC 114
3.6.5 Disable IPv6 115
3.6.6 Windows operating system firewall 118
3.6.7 IBM Spectrum Scale traces auxiliary tools 120
3.6.8 Installing Cygwin 121
3.6.9 Install IBM Spectrum Scale 121
3.6.10 Configure IBM Spectrum Scale embedded remote shell 128
3.6.11 Create IBM Spectrum Scale cluster 129
3.6.12 Creating NSD disks 130
3.6.13 Start the IBM Spectrum Scale cluster 132
3.6.14 Quorum configuration 133
3.6.15 Start IBM Spectrum Scale at boot 134

3.6.16 Create the IBM Spectrum Scale file system 134
3.6.17 Adding a Linux node to the Windows operating system cluster 135
3.6.18 Adding a Windows operating system node to the cluster 148
3.7 Oracle Real Application Cluster with IBM Spectrum Scale 152
3.8 IBM Spectrum Scale integration with IBM Spectrum Protect (formerly Tivoli Storage Manager) 159
3.9 Sample Spectrum Scale FPO configuration 170
3.10 Integrating with OpenStack Swift 175

Chapter 4. Management and maintenance 177
4.1 IBM Spectrum Scale and GPFS migration and update 178
4.1.1 GPFS and Spectrum Scale migration considerations 178
4.1.2 Migrating to Spectrum Scale 4.1 from GPFS 3.5 (Rolling update) 179
4.1.3 Migrating to Spectrum Scale 4.1 from GPFS 3.4 or GPFS 3.3 180
4.1.4 Migrating to Spectrum Scale 4.1 from GPFS 3.2 or earlier releases of GPFS . . 182
4.1.5 Completing the GPFS or IBM Spectrum Scale migration 183
4.1.6 Applying corrective fixes to IBM Spectrum Scale 185
4.2 Managing IBM Spectrum Scale cluster 187
4.2.1 Managing cluster repository 187
4.2.2 Changing cluster manager nodes 190
4.2.3 Managing IBM Spectrum Scale nodes 192
4.3 Managing IBM Spectrum Scale file systems 196
4.3.1 Listing file systems 197
4.3.2 Mounting a file system 198
4.3.3 Unmounting a file system 199
4.3.4 Creating an IBM Spectrum Scale file system 199
4.3.5 Removing a file system 201
4.3.6 Repairing a file system 201
4.3.7 Reducing file system fragmentation 204
4.3.8 Listing file system attributes 205
4.3.9 Changing file system attributes 206
4.3.10 Optimizing extended attributes: The fastea option 207
4.4 Managing IBM Spectrum Scale disks 209
4.4.1 Displaying disks 209
4.4.2 Creating NSDs 209
4.4.3 Adding a disk to a file system 212
4.4.4 Deleting disks from a file system. 214
4.4.5 Replacing disks in an IBM Spectrum Scale file system 215
4.5 Managing IBM Spectrum Scale data migration 216
4.5.1 Migrating data using NSDs in GPFS file systems 216
4.5.2 Migrating data using AFM-based NFS Migration 219
4.6 Managing the IBM Spectrum Scale network 221
4.7 Managing IBM Spectrum Scale remote cluster 223
4.7.1 Planning and preparation 223
4.7.2 Adding a remote cluster 224
4.7.3 Enabling GPFS replication 228
4.7.4 Exporting or importing a file system 232
4.8 Configuring IBM Spectrum Scale callback 238
4.9 Monitoring IBM Spectrum Scale with SNMPD protocol. 240
4.10 SSH configuration 242

Chapter 5. Information lifecycle management 245
5.1 Explaining the ILM concept 246

5.2 Fileset 246
5.3 Snapshot 251
5.3.1 Snapshot at the file system level 251
5.3.2 Snapshot at the fileset level 256
5.3.3 Snapshot at file level 259
5.4 Storage pools 260
5.4.1 Internal storage pools 260
5.4.2 External storage pools 262
5.5 Immutability and appendOnly attributes 265
5.6 Quotas 267
5.6.1 Enabling and editing quotas 267
5.6.2 Creating quota reports 269
5.7 Policies and rules 270
5.7.1 Policies 270
5.7.2 Rules 271
5.7.3 Example of polices with internal pools only 271
5.7.4 Example of policies with one external pool only 277
5.8 Access control list 280
5.8.1 Traditional Spectrum Scale ACL administration 280
5.8.2 NFSv4 ACL administration 283
5.9 The mmfind policy sample 285
5.10 Differences with IBM EasyTier 288

Chapter 6. Active File Management 291
6.1 Active file management fundamentals 292
6.1.1 AFM mode 293
6.1.2 Minimum requirements and basic operating system tuning 294
6.1.3 Using NFS for data movement 296
6.2 AFM single-writer 298
6.2.1 Creating a single-writer Cache fileset 299
6.2.2 Working with AFM filesets (single-writer) 300
6.2.3 Running single-writer fileset in disconnected mode 302
6.2.4 Auto recovering from failed connections 304
6.2.5 Manual recovering from failed connections 305
6.2.6 Permanent loss of HOME for single-writer 306
6.2.7 Permanent loss of Cache (single-writer) 308
6.2.8 Summary for AFM with single-writer 310
6.3 AFM independent-writer 311
6.3.1 Steps to create AFM enabled fileset in independent-writer mode 311
6.3.2 Working with independent-writer filesets 313
6.3.3 Cache eviction in independent-writer mode 316
6.3.4 Running independent-writer fileset in disconnected mode 319
6.3.5 Manual recovery from failed connections 320
6.3.6 Permanent loss of HOME 322
6.3.7 Temporary loss of Cache 322
6.3.8 Using AFM independent-writer for DR scenarios 324
6.3.9 Summary 330
6.4 Using AFM to migrate the content of an existing file system 330
6.5 Customizing and tuning AFM filesets 335
6.6 Building a global name space 337
6.7 Enhancements in IBM Spectrum Scale 4.1 TL 1 338
6.7.1 Gateway node changes 338
6.7.2 Native GPFS protocol 339

6.7.3 Performing parallel data transfers 341

Chapter 7. Backup and disaster recovery using IBM Spectrum Scale 345
7.1 Disaster recovery solution using IBM Spectrum Scale replication 346
7.1.1 Configuration 346
7.1.2 Characteristics of this DR configuration 347
7.1.3 The IBM Spectrum Scale mmfsctl command 348
7.2 Implementing a scenario with IBM Spectrum Scale replication 348
7.2.1 Environment: Hardware, software, network, storage 348
7.2.2 IBM Spectrum Scale configuration 348
7.2.3 IBM Spectrum Scale configuration diagram 350
7.2.4 Set up and configure the IBM Spectrum Scale DR cluster 351
7.3 Backup and restore for IBM Spectrum Scale 373
7.3.1 mmbackup utility 373
7.3.2 Spectrum Scale advanced backup tools 383
7.3.3 Scale Out Backup and Restore 388

Chapter 8. Problem determination 411
8.1 Problem determination process 412
8.1.1 Understanding the problem 412
8.1.2 Gathering information from the user 412
8.2 Gathering system information 413
8.2.1 Gathering operating system and IBM Spectrum Scale information 414
8.2.2 Testing connectivity and remote access 415
8.3 Operating system logs and IBM Spectrum Scale messages 415
8.3.1 Operating system logs 415
8.3.2 IBM Spectrum Scale messages 416
8.4 Verifying IBM Spectrum Scale cluster status 419
8.4.1 The mmgetstate command 419
8.4.2 The mmlsconfig command 420
8.4.3 The mmlscluster command 420
8.5 Collecting IBM Spectrum Scale file system and disk information 421
8.5.1 The mmlsfs command 421
8.5.2 The mmlsmount command 422
8.5.3 The mmlsnsd command 423
8.5.4 The mmwindisk command 424
8.5.5 The mmpmon command 424
8.6 Collecting IBM Spectrum Scale general information and logs 427
8.6.1 Considerations when gathering Spectrum Scale information 428
8.6.2 IBM Spectrum Scale severity tags 429
8.6.3 Spectrum Scale logs 430
8.6.4 The gpfs.snap command 433
8.6.5 Using the gpfs.snap command 436
8.7 Collecting IBM Spectrum Scale debug information 439
8.7.1 Collecting specific information 439
8.8 Working with IBM Spectrum Scale trace facility 445
8.8.1 Generating Spectrum Scale tracing information 446
8.8.2 Trace data analysis commands and scripts 448
8.9 Deadlock detection features 450
8.9.1 Automated deadlock detection 450
8.9.2 Automated deadlock data collection 451
8.9.3 Automated deadlock breakup 452
8.10 Additional information 452

8.11 IBM Spectrum Scale problem determination scenarios 453
8.11.1 Scenario 1: Spectrum Scale daemon not running. 453
8.11.2 Scenario 2: Analyzing waiters 455
8.11.3 Scenario 3: Spectrum Scale file system hang. 458
8.11.4 Scenario 4: Spectrum Scale file system unmounting 461
8.11.5 Scenario 5: Spectrum Scale file system not mounting 464
8.11.6 Scenario 6: Upgrading GPFS or Spectrum Scale 467
8.11.7 Scenario 7: NSD failures. 468

Chapter 9. Encryption 473
9.1 Introduction to encryption with IBM Spectrum Scale. 474
9.2 How encryption is implemented in IBM Spectrum Scale. 475
9.2.1 Support statement for encryption with other IBM Spectrum Scale features 476
9.3 Step-by-step setup procedure 477
9.3.1 Installing and setting up the IBM Security Key Lifecycle Manager 477
9.3.2 Installing IBM Security Lifecycle Manager 478
9.3.3 Accessing KLM 480
9.3.4 Updating the server. 480
9.3.5 Creating certificates and keys for use with IBM Spectrum Scale and IBM Security Lifecycle Manager. 481
9.3.6 Prepare IBM Security Lifecycle Manager for use with IBM Spectrum Scale. . . . 489
9.3.7 Configure IBM Spectrum Scale for encryption 493
9.4 Working with encryption 498
9.4.1 Verify encryption 498
9.4.2 Back up your keys. 499
9.4.3 Remote key manager and RKM.conf 500
9.4.4 Secure deletion file data 500
9.4.5 Understanding read access to an encrypted file. 504
9.5 Implementing access control on a node base. 505
9.5.1 Changes to RKM. 506
9.5.2 Changes to RKM.conf 507
9.5.3 Changes to Spectrum Scale policy 509
9.5.4 Activating access control 510
9.5.5 Rewrap FEKs to achieve node isolation from data 512
9.5.6 Securing your environment 513
9.5.7 Summary. 513

Appendix A. Recovery of the IBM Spectrum Scale file system configuration 515
Back up the file system configuration. 516
Generating the scripts for creating the original file system configuration. 518
Restoring the file system configuration 520

Appendix B. How to obtain an IBM Spectrum Scale trial version 525

Related publications 527
IBM Redbooks 527
Other publications 527
Online resources 527
Help from IBM 528

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2015. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Active Memory™
AFS™
AIX®
AIX 5L™
BigInsights™
DB2®
DS8000®
Easy Tier®
ECKD™
FlashCopy®
FlashSystem™
GPFS™
IBM®
IBM FlashSystem®
IBM Spectrum™
IBM Watson™
IBM z™
InfoSphere®
Linear Tape File System™
Micro-Partitioning®
NetView®
Passport Advantage®
POWER®
Power Systems™
POWER7®
POWER7+™
POWER8™
PowerHA®
PowerVM®
PureFlex®
pureScale®
Redbooks®
Redpaper™
Redbooks (logo) ®
RS/6000®
SANergy®
Storwize®
Symphony®
System i®
System p®
System Storage®
System z®
Tivoli®
WebSphere®
XIV®
z/VM®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Linear Tape-Open, LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

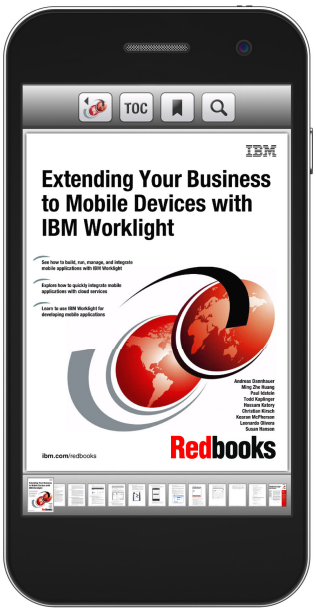
Other company, product, or service names may be trademarks or service marks of others.

Find and read thousands of IBM Redbooks publications

- Search, bookmark, save and organize favorites
- Get up-to-the-minute Redbooks news and announcements
- Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App

iOS **Download Now** Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!

ibm.com/Redbooks
About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

This IBM® Redbooks® publication updates and complements the previous publication: *Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment*, SG24-7844, with additional updates since the previous publication version was released with IBM General Parallel File System (GPFS™). Since then, two releases have been made available up to the latest version of IBM Spectrum™ Scale 4.1. Topics such as what is new in Spectrum Scale, Spectrum Scale licensing updates (Express/Standard/Advanced), Spectrum Scale infrastructure support/updates, storage support (IBM and OEM), operating system and platform support, Spectrum Scale global sharing - Active File Management (AFM), and considerations for the integration of Spectrum Scale in IBM Tivoli® Storage Manager (Spectrum Protect) backup solutions are discussed in this new IBM Redbooks publication.

This publication provides additional topics such as planning, usability, best practices, monitoring, problem determination, and so on. The main concept for this publication is to bring you up to date with the latest features and capabilities of IBM Spectrum Scale as the solution has become a key component of the reference architecture for clouds, analytics, mobile, social media, and much more.

This publication targets technical professionals (consultants, technical support staff, IT Architects, and IT Specialists) responsible for delivering cost effective cloud services and big data solutions on IBM Power Systems™ helping to uncover insights among clients' data so they can take actions to optimize business results, product development, and scientific discoveries.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Dino Quintero is a complex solutions Project Leader and an IBM Senior Certified IT Specialist with the ITSO in Poughkeepsie, NY. His areas of knowledge include enterprise continuous availability, enterprise systems management, system virtualization, technical computing, and clustering solutions. He is an Open Group Distinguished IT Specialist. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

Luis Bolinches has been working with IBM Power Systems servers for over 10 years and has been with GPFS for six years. He worked three years at CERN, and has been an IBMer in Spain, Estonia, and currently in Finland. He has several years of experience with Intel and Power Systems servers, and GPFS clusters as well.

Puneet Chaudhary is a Technical Solution Architect with the GPFS and TC Solutions Enablement team.

Willard Davis is a Software Test Specialist with the GPFS development and testing team.

Steve Duersch is a GPFS test manager. He joined IBM in 2001. Steve has been a GPFS test lead for four years and manager for two years.

© Copyright IBM Corp. 2015. All rights reserved.

Carlos Henrique Fachim is a Field Technical Sales Specialist working in the Systems & Technical Group in IBM Brazil for pre-sales technical advisory and architecture designs. He joined IBM 16 years ago, working as an IBM AIX® and an IBM RS/6000® Support Specialist. His technical expertise areas include AIX (also DUMP and Performance Analysis), IBM PowerHA®, GPFS, Power Systems and IBM PowerVM® features, SAN, and IBM System Storage® products (V7000, IBM DS8000®, SAN Volume Controller, and Tivoli Storage Productivity Center). He also has much experience designing and implementing highly virtualized and highly available environments. He is a member of the IBM Technology Leadership Council Brazil, and holds a Master's degree of Computer Science from Mackenzie University (Sao Paulo, Brazil). He is a Certified Advanced Technical Expert Specialist.

Andrei Socoliuc is a Certified IT Specialist in Systems and Infrastructure, working in IBM Global Technologies Services Romania. He has more than 12 years of experience in IT infrastructure. Andrei holds a Master's degree in Computer Science from the Polytechnical University of Bucharest. He is a Certified Advanced Technical Expert IBM Power Systems and a Certified Tivoli Storage Manager specialist. He is also a coauthor of several IBM PowerHA IBM Redbooks publications.

Olaf Weiser is an IT Infrastructure Specialist focused primarily on integrating Linux, AIX, and cluster deployments into enterprise environments. During his career at IBM, he has worked with multiple product families and has had extensive experience designing, deploying, testing, and auditing enterprise architectures for automotive and telecommunication customers. His current technical work covers clusters in HPC environments, as well as managing classic IT environments. Of particular importance is the integration of business requirements and IT capabilities, by assisting clients by making tactical and strategic recommendations for their IT infrastructure. Olaf has a degree in Civil Engineering from Saxon University of Cooperative Education (State Academy Saxon).

Thanks to the following people for their contributions to this project:

Richard Conway
IBM International Technical Support Organization, Poughkeepsie Center

April L. Brown, Manoj Naik
IBM US

Kalyan Gunda
IBM India

Bruno Blanchard
IBM France

Marcelo Scabim
IBM Brazil

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  https://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

The new IBM Spectrum Scale is a proven, scalable, high-performance data and file management solution (formerly IBM General Parallel File System or GPFS).

The IBM Spectrum Scale is the new name for version 4.1. Before version 4.1, the name GPFS will be maintained.

This book provides information about new features, configuration improvements, and general enhancements added in Spectrum Scale version 4.1. Also, it incorporates features added in GPFS 3.5 and IBM Spectrum Scale after the publication of *Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment*, SG24-7844.

This chapter contains the following topics:

- Overview
- IBM Spectrum Scale major components and terminology
- IBM Spectrum Scale new features and enhancements
- IBM Spectrum Scale competitive strategy
- IBM Spectrum Scale licensing
- IBM Spectrum Scale on cross-platform environments
- IBM Spectrum Scale and virtualization
- Contact information

© Copyright IBM Corp. 2015. All rights reserved.

1.1 Overview

The top two challenges that organizations face with IT infrastructure are storage related: Data management and cost efficiency[1].

For enterprises that are swamped by unstructured data, IBM Spectrum Scale software lets you share the storage infrastructure while it automatically moves file and object data to the optimal storage tier as quickly as possible, that is, move data automatically between flash, disk, and tape.

Spectrum Scale enjoys highly differentiated value:

- Virtually limitless scaling to nine quintillion files and yottabytes of data.
- Provides high performance, over 400 GBps, and simultaneous access to a common set of shared data.
- Over 400 GBps performance.
- Software Defined Storage lets you build your infrastructure your way:
  - Easy to scale with relatively inexpensive commodity hardware while maintaining world-class storage management capabilities.
  - Use any combination of flash, spinning disk, and tape.
  - Use various cluster models that include storage area networks (SANs), Network Shared Disk, and Shared Nothing clusters.
  - Add more storage capacity without affecting the application to greatly simplify administration.
- Integrated information lifecycle management (ILM) Tools automatically move data based on policies. This can dramatically reduce operational costs as fewer administers can manage larger storage infrastructures.
- Provides global data access across geographic distances and unreliable WAN connections.
- Proven reliability with use in the most demanding commercial applications.
- Protects data from most security breaches, unauthorized access, or being lost, stolen, or improperly discarded with native file encryption for data at rest and secure erase.

Spectrum Scale is intended to be used by diverse workloads where performance, reliability, and availability of data are essential to the business. Spectrum Scale has being used extensively for many years across multiple industries worldwide for the world's most demanding data-intensive industries such as the following examples:

- Engineering design
- Media and Entertainment: Radio and TV
- Petroleum: Seismic data ingest and analytics
- Smarter cities: Video Surveillance
- Automotive: Crash test recording, autonomous driving, and so on
- Defense/Military: Flight recording
- Satellite image archival
- Telecom: Call detail records
- Banking and Financial Services Industry: Report, statements, check truncation
- Business intelligence
- Financial analytics

[1] "Source: IBM Institute for Business Value, The IT Infrastructure Conversation: New content, new participants, new tone, 2014"

- Data mining
- Scientific research
- Cognitive applications such as IBM Watson™

Some additional Spectrum Scale capabilities and benefits are:

- Increases resource and operational efficiency by pooling redundant isolated resources.
- Intelligent resource utilization and automated management of storage reduces storage costs and drives operational efficiencies and is able to use tiering capabilities to store the data and place the data efficiently.
- Multiple configuration options to provide optimal performance, flexibility, and reliability to eliminate single points-of-failure and automation to provide fast failover in a disk or server malfunction.
- Provides multi-site support, connecting your local Spectrum Scale cluster to remote clusters to provide Disaster Recovery configurations.
- Cross-platform solution enables you to work with different platforms and operating systems. It is possible to create a Spectrum Scale cluster using AIX, Linux, and Windows server nodes, or a mix of all three. Spectrum Scale is now available for IBM System z®.
- Achieve greater IT agility by being able to quickly react, provision, and redeploy resources in response to new requirements. Spectrum Scale is able to add or delete disks while the file system is mounted.

1.2 IBM Spectrum Scale major components and terminology

IBM Spectrum Scale has some unique components that we briefly describe in this section to help with the understanding of the next chapters and sections if you are unfamiliar with the Spectrum Scale structure terminology.

The Spectrum Scale has the following components:

- **Cluster**

  This consists of a number of nodes and network shared disks (NSDs) for management purposes. It can be configured to use server-based repository type, where the cluster needed to be explicitly configured to have a primary and secondary server to keep cluster configuration files or, starting in Spectrum Scale 4.1, a new repository type called *Cluster Configuration Repository* (CCR). This is where the configuration files are automatically maintained in all quorum nodes. More information about CCR is available under 4.2.1, “Managing cluster repository” on page 187.

- **Node**

  A node is any server that has the Spectrum Scale product installed with direct storage access or network access to another node. Depending on the type of access, each node can have different roles within the cluster configuration.

- **Cluster manager**

  The cluster manager node has overall responsibility for correct operation of all the nodes and the cluster as a whole. The cluster manager performs the following tasks:

  – Monitors disk leases

  – Detects failures and manages recovery from node failure within the cluster.

  – The cluster manager determines whether or not a quorum of nodes exists to allow the Spectrum Scale daemon to start and for file system usage to continue.

- Responsible to handle configuration information and to share configuration changes that must be known to nodes in remote clusters.
- Selects the *file system manager* node.

By default, the cluster manager is chosen through an election held among the set of quorum nodes designated for the cluster, but starting in GPFS 3.5, you can now explicitly define the nodes that can become the cluster manager node. More information is covered in section 4.2, “Managing IBM Spectrum Scale cluster” on page 187.

- **File system manager**

  This node maintains the availability information for the disks in the file system. In a large cluster, this might need to be a dedicated node that is separate from the disk servers. The file system manager performs the following tasks:

  - Manages file system configuration
  - Manages disk space allocation
  - Manages quota configuration
  - Handle security services

- **Network shared disk (NSD)**

  This component is used for global device naming and data access in a cluster. If all nodes do not have a direct connection to the disks (for example, in a SAN environment), an NSD must be defined with a primary server and it is highly recommended to have a secondary server also defined. I/O is then performed using the network connection to get to the NSD server that performs the I/O on behalf of the requesting node. Even if all NSDs are attached to the disks, it is still recommended to have NSD servers defined to provide a backup path if the primary server node lost the access to the physical disk.

- **Storage pool**

  Is a collection of NSDs and it is used to partition storage based on characteristics such as performance, locality, and reliability. The use of storage pools in Spectrum Scale allows you to group storage devices based on performance, locality, or reliability characteristics within a file system.

- **Block**

  A block is the largest unit for single I/O operation and space allocation in a Spectrum Scale file system. The block size is specified when a file system is created. The block size defines the stripe width for distributing data on each disk used by Spectrum Scale. Spectrum Scale supports block sizes ranging 16 KB - 16 MB and defaults to 256 KB in previous GPFS versions and 64 K if using Spectrum Scale at 4.1.0.4 level. Spectrum Scale allows different block sizes for metadata and data, if disks for data and metadata are kept separate.

- **Chunk**

  The term chunk is related to the Spectrum Scale *File Placement Optimizer* (FPO) feature. Chunk is a logical grouping of blocks that behaves like one large block. A block group factor dictates the number of blocks that are laid out on the disks attached to a node when using an FPO configuration to form a chunk. The file data is divided into chunks, and each chunk is mapped to a node according to write-affinity depth setting. A chunk is then mapped to all available disks within a node. The chunk size is defined by multiplying the block group factor and the block size. The block group factor ranges 1 - 1024. The default block group factor is 1 for compatibility with standard Spectrum Scale file systems. Setting the block size to 1 MB and the block group factor to 128 results in a chunk size of 128 MB. More details about FPO are covered in 3.9, “Sample Spectrum Scale FPO configuration” on page 170.

- **Failure group**

  A failure group is a set of disks that share a common point of failure that could cause them all to become simultaneously unavailable. When creating multiple replicas of a given block, Spectrum Scale uses failure group information to ensure that no two replicas exist within the same failure group. A failure group can be specified as a list of up to three comma-separated numbers that can help you to identify topology information.

- **Metanode**

  A node that handles metadata, also referred to as “directory block updates”.

- **Metadata**

  It contains specific cluster configuration information and non-user data.

- **Application node**

  This mounts a Spectrum Scale file system and runs a user application that accesses the file system.

- **Quorum nodes**

  These are nodes participating in maintaining the Spectrum Scale cluster active. There are two types of cluster quorum nodes:

  – *Node quorum*, where the cluster is maintained online when most of quorum nodes are available.

  – *Node quorum with tiebreaker disks*, where the cluster is online when one quorum node is up and it has access to the tiebreaker disks.

1.2.1 IBM Spectrum Scale cluster topologies

IBM Spectrum Scale provides various configurations for customer solutions. The following four configuration types are sample IBM Spectrum Scale solutions, which are classified by the location of the application on the cluster nodes:

- Application running on Spectrum Scale NSD clients only (network-based Spectrum Scale client model).
- Application running on nodes with direct attachment to the storage (direct attached storage model).
- Application running on both NSD servers and clients (mixed NSD access model).
- Application running in an IBM Spectrum Scale File Placement Optimizer cluster (IBM Spectrum Scale FPO model).

For a more detailed explanation of the Spectrum Scale cluster topologies, refer to 2.1, “IBM Spectrum Scale cluster topologies” on page 30.

1.3 IBM Spectrum Scale new features and enhancements

The IBM Spectrum Scale 4.1 has enhanced security features, flash accelerated performance, and improved usability for broad adoption. The licensing structure was also changed to provide more flexibility depending on the Spectrum Scale usage. More information about licensing is available under section 1.5, “IBM Spectrum Scale licensing” on page 12.

In the next topics, we show some of the new Spectrum Scale enhancements and features currently available, organized in the following subsections:

- Enhanced security
- Performance improvements
- New usability features
- Reliability, availability, and serviceability

The features are briefly described in this section and covered in this publication, or referred to the most proper source.

1.3.1 Enhanced security

The Spectrum Scale 4.1 encryption is designed to protect data on a disk from most security breaches, unauthorized access, or being lost, stolen, or improperly discarded. In this section, we describe the new security features:

- **Encryption of stored data**

  Spectrum Scale 4.1 Advanced Edition uses cryptographic erase, intended for fast, simple, and secure drive retirement for a multi-cluster environment and is designed for both secure storage and secure deletion for the stored data.

  Secure stored data

  - Each file is encrypted with a random *File-encrypting key* (FEK) that is controlled by one or more *Master encryption keys* (MEKs).
  - Encryption of data in the disk, the files are encrypted before they are stored on disk.
  - The encryption keys are never stored in the disk to avoid data leakage in case the physical disks are stolen or improperly decommissioned.

  Secure deletion

  - No data overwriting because each file has unique encryption keys.
  - Secure deletion is a cryptographic operation, when a standard file system operation is issued (rm, unlink for example) the encryption algorithm controls which files should be removed and after the operation is completed the MEK is updated.

- **Compliant** with *NIST Special Publication 800-131A* - **"***Recommended Security Controls for Federal Information Systems and Organizations*,*"* which specifies recommended encryption methods by implementing appropriate key management procedures.

Notes: Information about the *National Institute of Standards and Technology (NIST) Special Publication 800-131A* can be found at the following site:

`http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf`

1.3.2 Performance improvements

Starting in GPFS 3.5 some new performance features have been added to accelerate parallel and shared data access and handle the most demanding data-intensive applications worldwide. Some of the new performance improvements include:

- ***Active File Management* (AFM).** AFM was introduced in GPFS 3.5 and is a scalable, high-performance, file system caching layer integrated with the Spectrum Scale cluster file. It enables the sharing of data across unreliable or high latency networks for long-distance clusters. It creates associations from a local Spectrum Scale cluster to a remote cluster or storage, defining the location and flow of file data and automating the

data management. AFM masks wide area network latencies and outages by using Spectrum Scale to cache massive data sets, allowing data access and modifications even when remote storage cluster is unavailable. In addition, AFM performs updates to the remote cluster asynchronously, which allows applications to continue operating while not being constrained by limited outgoing network bandwidth, making the cache completely transparent to applications.

- **The AFM parallel read and write**. In Spectrum Scale 4.1, the AFM performance was improved, allowing Spectrum Scale to read or write large blocks of data in a single I/O operation, thereby minimizing overhead. This improvement provides high-performance I/O by "striping" blocks of data from individual files across multiple disks (on multiple storage devices) and reading and writing these blocks in parallel.

  Parallel reads

  – Multiple gateway nodes can be used to prefetch a single file, which is designed to improve performance.

  Parallel writes

  – Multiple gateway nodes can be used to synchronize changes to small and large files, also designed to improve I/O performance.

- ***Highly Available Write Cache*** **(HAWC)**. This feature enables the Spectrum Scale to place the Spectrum Scale recovery log in NVRAM (such as SSDs or flash-backed memory) to reduce the latency of small and synchronous writes. It can benefit applications that experience bursts of small and synchronous write requests. Basically there are two methods of using HAWC:

  – Store Spectrum Scale recovery log in a highly available and fast data store (as SSDs) in a storage controller or on a separate SSD box (such as flashsystems 840).

  – Store and replicate Spectrum Scale recovery log in unreliable NVRAM in the Spectrum Scale client nodes.

- ***Local Read Only Cache*** **(LROC)**. This is a new option for Linux performance improvement to support large local read-only cache using solid-state disks. Many applications can benefit greatly if the response time from the storage subsystem is faster. When using LROC, the data is available with very low latency and the cache serves to reduce the load on the shared network and on the backend disk storage, increasing application performance, while still using the benefits of shared storage. Not only is the data available faster, but the cache hit serves to reduce the load on the shared network and on the backend storage.

- **Network performance improvements**. Spectrum Scale 4.1 provides network performance monitoring in the form of RPC latency measurement to detect and troubleshoot networking issues that might affect Spectrum Scale operation. Optimal Spectrum Scale performance depends on proper network performance and behavior.

1.3.3 New usability features

The new GPFS version 3.5 and Spectrum Scale 4.1 also provide more flexibility, management, and control of resources by adding the following features:

- ***GPFS Storage Server Version*** **(GSS):** GSS was introduced in GPFS 3.5 and is a high-capacity, high-performance storage solution that combines Spectrum Scale software, using the GPFS native raid (*GNR*) implementation and a dedicated storage solution. More details about GSS are described in "IBM Spectrum Scale Server solution" on page 17.

- **AFM operation:** Spectrum Scale 4.1 includes features to optimize the AFM operation, including:
  - Prefetch enhancements that prepopulate the AFM cache so file data is resident locally before it is directly requested by an application.
  - AFM now supports the native Spectrum Scale protocol for the AFM communication channel, providing for improved integration of Spectrum Scale features and attributes.
  - AFM was improved for fast recovery in a gateway node failure, introducing a new version of hashing (afmHashVersion=2), which minimizes the impact of gateway nodes joining or leaving the active cluster.
- **AFM-based NFS data migration:** Spectrum Scale Data migration eases data transfer when upgrading hardware or buying a new system and it can minimize downtime for applications. Some of the benefits on this migration option are as follows:
  - Move actual file data along with any permissions and ACLs associated with files from the current system to the new Spectrum Scale system.
  - Consolidate data from multiple earlier systems keeping associated configuration data intact.
  - Data migration can be done by Volume/Block/Disk level migration, backup and restore, and other methods.
- **Cygwin replaces SUA for Windows system nodes:** SUA is no longer supported for Windows systems nodes. Cygwin is now a prerequisite before installing Spectrum Scale on Windows operating system nodes.
- **Backup and restore improvements:** These include some new features and tools to provide faster options for multiple backups of data and security enhancements. Some of the features are:
  - **Fileset snapshot restore:** This is a tool designed to restore data from a fileset snapshot into the active file system. The tool can deal with files that have been removed, added, or changed since the snapshot was taken. Extended attributes saved at the snapshot are also restored.
  - **Snapshot command batching:** This is a new feature introduced in Spectrum Scale 4.1 that allows the combination of multiple snapshot operations at the same time. Usually the snapshot commands require several global resources that can become points of contention. By combining multiple snapshots, the batching requires global resources less often and for shorter periods of time, consequently reducing the overhead and improving overall system throughput and performance during the snapshot process. More information about backup features is available in Chapter 7, “Backup and disaster recovery using IBM Spectrum Scale” on page 345.
  - **The mmbackup command:** This command was improved to allow tuning options for each node in the Spectrum Scale cluster. It now automatically adjusts its own work distribution and parallel access to *IBM Tivoli Storage Manage*r according to resources available and user-provided input parameters.
- ***RDMA over Converged Ethernet*** **(RoCE):** RoCE is a link layer protocol that provides efficient low-latency RDMA services over Ethernet layer. RDMA is used to transfer data directly between the *Network Shared Disks* (NSDs) Client pagepool buffer and the NSD Server pagepool buffer instead of sending and receiving the data over the TCP socket. The use of RDMA can improve bandwidth performance and decrease the use of operational resources. The **RDMA** or *Remote Direct Memory Access* is a protocol that provides read and write services directly to applications without using the operating system layer.

- **Clustered NFS (cNFS) support:** Spectrum Scale now supports cNFS V4 on *SUSE Linux Enterprise Server* (SLES) and *Red Hat Enterprise Linux* (RHEL). Spectrum Scale has been enhanced to support IPv6 and NFS V4.
- **IBM Spectrum Scale for Linux on System z:** Spectrum Scale 4.1 now is available in a System z machine running Linux. This new feature enables enterprise clients to use a highly available clustered file system with Linux in a logical partition (LPAR) or as a Linux guest on IBM z™ machine.
- **NSD enhanced format:** A new NSD format was introduced with Spectrum Scale 4.1. The new format is referred to as NSD v2, and the old format is referred to as NSD v1. The NSD v1 format is compatible with GPFS releases before 4.1. The new Spectrum Scale recognizes both NSD v1 and NSD v2 formatted disks. Some of the NSD version 2 benefits are as follows:
  - Adjusts data alignment to support disks with a 4 KB physical block size
  - Adds backup copies of some key Spectrum Scale data structures
  - Expands some reserved areas to allow for future growth
- ***GUID Partition Table* (GPT) support:** This Spectrum Scale enhancement adds a standard GPT to the device when creating NSDs on Linux operating systems. This feature was introduced by the NSD version 2 and avoids direct attach NSDs to appear as unused devices, what could lead the administrators to inadvertently corrupt Spectrum Scale data.
- ***IBM Spectrum Scale File Placement Optimizer* (FPO)**: This feature enables each node to operate independently, reducing the impact of failure events across multiple nodes. It allows Spectrum Scale to use locally attached disks on a cluster of servers communicating using the network, rather than the regular case of using dedicated servers for shared disk access (such as using SAN). Spectrum Scale FPO is suitable for workloads like *SAP HANA* and IBM DB2® with Database Partitioning Feature, and it can be used as an alternative to Hadoop Distributed File System (HDFS) in big data environments. The use of FPO extends the core Spectrum Scale architecture, providing greater control and flexibility to leverage data location, reduce hardware costs, and improve I/O performance. Some benefits when using FPO are:
  - Allows your jobs to be scheduled where the data resides (*Locality awareness*).
  - Metablocks that allow large and small block sizes to coexist in the same file system.
  - Write affinity that allows applications to dictate the layout of files on different nodes, maximizing write and read bandwidth.
  - Pipelined replication to maximize use of network bandwidth for data replication.
  - Distributed recovery to minimize the effect of failures on ongoing computation.

More information about IBM Spectrum Scale FPO can be found in the following resources:

- *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00

  http://publibz.boulder.ibm.com/epubs/pdf/c2370320.pdf
- *IBM System x Reference Architecture for Hadoop: IBM InfoSphere BigInsights Reference Architecture*, REDP-5009

  http://www.redbooks.ibm.com/abstracts/redp5009.html?Open
- Deploying a big data solution using IBM GPFS-FPO

  http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF

1.3.4 Reliability, availability, and serviceability

Starting in GPFS 3.5, some of the new IBM Spectrum Scale RAS enhancements include:

- Spectrum Scale provides more reliability and flexibility and now offers the option to select the nodes that can become cluster manager nodes in case of failures. Before Spectrum Scale 4.1, only a quorum node could be selected to become the cluster manager in case of cluster manager failure. A new option was added in Spectrum Scale 4.1 to allow the system administrators to change this behavior, More details are explained in 4.2.2, “Changing cluster manager nodes” on page 190.
- The **mmfsck** command was improved to repair only the data blocks that were changed during the time the disk was down, decreasing the time spent during the **mmfsck** operation.
- Spectrum Scale 4.1 introduces a new quorum-based repository for the configuration data. When cluster configuration repository (CCR) is enabled Spectrum Scale stores cluster metadata on all quorum nodes, instead of just the cluster primary and secondary nodes. This allows a more robust environment capable of withstanding more errors or failures.
- The new deadlock amelioration feature was added to improve data gathering during deadlock situations and to help system administrators to provide quick response when facing deadlock problems.
- The enhanced message logging in Spectrum Scale 4.1 allows messages to be sent to syslog on Linux. Severity tags were added to numerous messages, and these tags can be used to filter the messages that are sent to syslog. This messaging format produces precise information regarding some specific failures, action of informative messaging.
- Starting on GPFS 3.5, the NSD configuration is improved to provide more tunable options for NSD threads. Basically, before GPFS 3.5 the NSD server had a single queue to handle requests from the clients. Starting on GPFS 3.5, the NSD Server has multiple thread queues to handle requests. A new tool is available to monitor the usage of the NSD threads. More details are provided under the “Collecting IBM Spectrum Scale debug information” on page 439.

Note: For a full list of changes, check the *GPFS V4.1: Concepts, Planning, and Installation Guide*, GA76-0441-00, which is available at the following link:

`http://publib.boulder.ibm.com/epubs/pdf/a7604410.pdf`

1.4 IBM Spectrum Scale competitive strategy

Spectrum Scale is positioned as software-defined storage for cloud, big data, and analytics, for applications that demand high performance and shared access to a common set of local or remote data and files. The combination of high availability, performance, and management is perfect for the most critical applications and workloads. The Spectrum Scale can be used for any type of workload and it is not designed only for high performance computing.

Some unique differentiation points for Spectrum Scale versus other files systems are as follows:

- Spectrum Scale is not a client/server model such as NFS, CIFS, IBM AFS™, or DFS. Spectrum Scale is not a single-server bottleneck, because when using Spectrum Scale you can have multiple servers holding and sharing the data.
- One metanode exists per open file. The *metanode* is responsible for maintaining file metadata integrity. In almost all cases, the node that has had the file open for the longest

continuous time is the metanode. All nodes accessing a file can read and write data directly, but updates to metadata are written only by the metanode.

- Spectrum Scale commands can be executed from any node in the cluster. If tasks must be performed on another node in the cluster, the command automatically redirects the request to the appropriate node for execution. For administration commands to be able to operate, passwordless remote shell communications between the nodes can be required depending on your cluster configuration. If needed, this task can be accomplished by using a standard shell program, such as **rsh** or **ssh**, or a custom application that uses the same semantics. To provide additional security, you can control the scope of passwordless access by allowing administration tasks to be performed from either of the following sources:
  - From all nodes running Spectrum Scale (by using **mmchconfig adminMode=allToall**)
  - From a subset of nodes (by using **mmchconfig adminMode=central**)
- Spectrum Scale consists of a single installation product and it does not require additional software for configuration. Other cluster file system solutions require additional software packages such as cluster server, volume manager, and file system software. This is a strong point, which helps reduce complexity for installation and administration of the cluster file system.
- Spectrum Scale provides multiple features that improve the reliability of your file system. This improved reliability includes automatic features such as file system logging, and configurable features such as intelligently mounting file systems on start and providing tools for flexible synchronous replication. Also, Spectrum Scale provides a heterogeneous infrastructure with AIX, Linux, and Windows operating systems. However, other cluster file systems do not support this kind of architecture, but do take advantage of CIFS and NFS with their own shared file system.

Some other Spectrum Scale features can be observed in Table 1-1.

Table 1-1 Comparing Spectrum Scale features with other similar products

Features	Spectrum Scale	Others
High performance support for traditional applications	Manages metadata by using the local node when possible rather than reading metadata into memory unnecessarily: ► Caches data on the client side to increase throughput of random reads. ► Supports concurrent and parallel reads and writes by multiple programs. ► Provides sequential access that enables fast sorts.	Some products provide either performance or scalability, cannot combine both in the same cluster, limited capacity to provide parallelism.
High availability	Has no single point of failure because the architecture supports the following attributes: ► Quorum nodes replicating configuration data. ► Data replication. ► Automatic distributed node failure recovery and reassignment in case of system or storage failures.	Limited high availability features, mostly for server failures only.

Features	Spectrum Scale	Others
POSIX compliance	Is fully POSIX-compliant, which provides the following benefits: ► Support for a wide range of traditional applications. ► Support for UNIX utilities that enable file copying by using FTP or SCP. ► Multiple NSD servers.	Mostly, products are not POSIX-compliant, which creates the following limitations: ► Limited support of traditional applications, after the initial load, data is read-only. ► Lucene text indexes must be built on the local file system or NFS because updating, inserting, or deleting entries is not supported.
Support for mixed workloads	A single Spectrum Scale cluster can support the requirements of a mix of analytics, databases, and other workloads.	Usually requires separate clusters for different workloads.
Data encryption	Supports data encryption within file and fileset level.	Limited or no encryption available.
Data replication	► Provides cluster-to-cluster replication over a wide area network. ► Provides replication via failure groups. Spectrum Scale can have multiple copies of the data and metadata within a single cluster.	Limited replication options.
Scalability	Spectrum Scale 4.1 current tested limits are (there are no hard limits at this point): ► Up to 9620 Linux nodes, 1530 AIX nodes, and up to 64 Windows system nodes. ► Disk sizes up to 2 TB. ► File system sizes up to 18 Petabytes. ► Can mount up to 256 file systems in a single cluster. ► Depending on file system, formats can have up to 9,000,000,000 files. ► Up to 2048 disks in a single file system.	Depending on the product, the limitations can be different. For more information about the current limits, consult specific vendor guides.
Supported platforms	AIX, Linux, Windows operating system	Mostly products are supported only on Linux platforms.

1.5 IBM Spectrum Scale licensing

The license metric for Spectrum Scale 4.1 has changed from the previous versions, where it was based on processor core, to a metric based on the number of sockets on the server, or

virtual server, available to Spectrum Scale. In a virtual environment, the number of Spectrum Scale licenses required by the virtual server is the sum of cores allocated to all the virtual machines using Spectrum Scale on the physical server divided by the number of cores per physical sockets, rounded up to a whole number if a fraction. For example, on a 2 socket, 10 cores/socket physical server, 6 cores are available to the VMs on that server. Therefore, the number of licenses required is equal to 6/10 (.6) rounded to 1.

The Spectrum Scale 4.1 is now available in three different editions (refer to Table 1-2):

- **Express Edition**: Contains the base Spectrum Scale functionality.
- **Standard Edition**: It is technically equivalent to GPFS 3.5 product. It includes the base functions plus *Information Lifecycle Management, Active File Management*, and *Clustered NFS*.
- **Advanced Edition**: Adds the *Encryption* function to the features of Standard Edition.

Table 1-2 IBM Spectrum Scale editions capabilities

Capabilities	Express Edition	Standard Edition	Advanced Edition
Multi-protocol scalable file service with simultaneous access to a common set of data	ES native client driver Hadoop connector	ES native client driver Hadoop connector NFS 3 or NFS 4.0	ES native client driver Hadoop connector NFS 3 or NFS 4.0
Facilitate data sharing with a global namespace, simplified management at scale (massively scalable file system, quotas, and snapshots) data integrity and availability	Included	Included	Included
Create optimized tiered storage pools by grouping disks based on performance, locality, or cost characteristics	Not included	Included	Included
Simplify data management at scale with information lifecycle management (ILM) tools that includes file sets, policy-based data placement and migration, backup and recovery, and archiving to a low-cost storage pool	Not included	Included	Included

Capabilities	Express Edition	Standard Edition	Advanced Edition
Enable worldwide data access and empower global collaboration using AFM asynchronous replication	Not included	Included	Included
Protect data with native encryption and secure erase, NIST compliant, and FIPS certified	Not included	Not included	Included

Important: The following considerations apply when using the Spectrum Scale editions:

- All nodes in a cluster must use the same edition.
- Spectrum Scale Advanced Edition is not supported on the Windows operating system.
- Licenses can be upgraded to a higher priced edition, but not downgraded.

The following Spectrum Scale installation packages are present in each edition (for AIX and Linux environments, the packages marked in bold differentiate between the editions):

- **Express Edition**: gpfs.base, gpfs.docs, gpfs.gpl, gpfs.msg, gpfs.gnr, gpfs.gskit
- **Standard Edition**: gpfs.base, gpfs.docs, gpfs.gpl, gpfs.msg, gpfs.gnr, gpfs.gskit, **gpfs.ext**
- **Advanced Edition**: gpfs.base, gpfs.docs, gpfs.gpl, gpfs.msg, gpfs.gnr, gpfs.gskit, gpfs.ext, **gpfs.crypto**

Each edition of Spectrum Scale 4.1 has three types of licenses:

- **Server license** is needed to perform the following functions:
  - Management functions such as cluster configuration manager, quorum node, manager node, and Network Shared Disk (NSD) server.
  - Sharing data directly through any application, service protocol, or method, such as Network File System (NFS), Common Internet File System (CIFS), File Transfer Protocol (FTP), or Hypertext Transfer Protocol (HTTP).
- **Client license** allows the exchange of data between servers that locally mount the same Spectrum Scale file system. No other export of the data is permitted. Spectrum Scale client-licensed servers can access the file system disk using the network via NSD servers or by using direct attachment (such as SAN), under the following circumstances:
  - Do not perform Spectrum Scale NSD server functions
  - Are not defined to the Spectrum Scale cluster as NSD servers
  - Do not perform Spectrum Scale cluster management functions
  - Do not share Spectrum Scale data directly through any application, service protocol, or method, like participating in quorum or being a file system manager
  - Do not access a Spectrum Scale-FPO file system
- **File Placement Optimizer license**: Permits the licensed servers to perform NSD server functions for sharing Spectrum Scale data with other servers that have a Spectrum Scale FPO or Spectrum Scale Server license. This license cannot be used to share data with

servers that have a Spectrum Scale Client license or non-Spectrum Scale servers. This server does not replace the server license.

Figure 1-1 shows a flow chart describing which IBM Spectrum Scale editions license to use depending on which purpose.

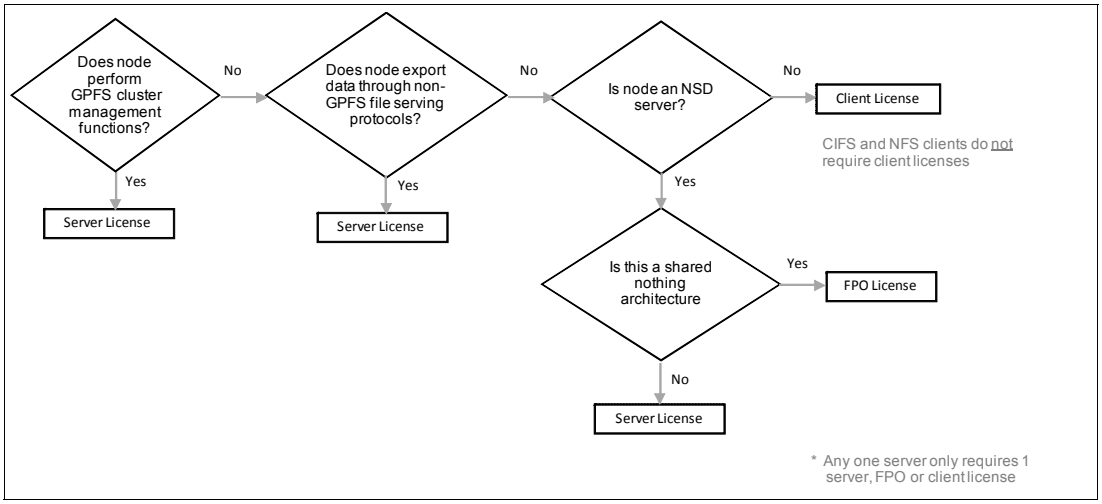


Figure 1-1 Flow chart to help decide which edition to buy

For more details about the purchase and licensing of Spectrum Scale, see:

- Spectrum Scale 4.1 announcement letter:

  http://ibm.co/19sithB

- Spectrum Scale frequently asked questions:

  http://ibm.co/1IKO6PN

Important: Regarding licensing virtual machines, some statements should be considered:

- For all nodes (LPARs) configured as uncapped partitions, the number of virtual processors must be considered.
- For all nodes (LPARs) configured as capped, the value of entitled capacity must be considered.

1.6 IBM Spectrum Scale on cross-platform environments

The Spectrum Scale version 4.1 is supported on multiple operating systems and hardware platforms, each one with specific characteristics and specifications. For more information about specific supported versions, fix levels, and mixed environments, see Chapter 2, “Infrastructure planning and considerations” on page 29.

1.6.1 IBM Spectrum Scale for AIX

Spectrum Scale for AIX is supported on IBM POWER® systems running AIX 6.1 or 7.1 in dedicated or shared logical partitions. Customers with Spectrum Scale version 4.1 must be running AIX 6.1.4.0 level or later.

AIX supports the following Spectrum Scale editions:

- Express Edition
- Standard Edition
- Advanced Edition
- File Placement Optimizer (FPO)

AIX environments also support Active File Management (AFM).

1.6.2 IBM Spectrum Scale for Windows operating systems

Spectrum Scale for Windows systems is supported in different Windows systems versions and editions:

- Windows Server 2008 x64 (SP2)
- Windows Server 2008 R2
- Windows Server 2012 (Datacenter and Standard)
- Windows Server 2012 R2 (Datacenter and Standard)
- Windows 7 (Enterprise and Ultimate editions) x64 SP1
- Windows 8.1 (Enterprise edition) in both heterogeneous (Windows and non Windows operating system nodes) and homogeneous (Windows system-only nodes) clusters.

Currently, Windows operating systems support the following Spectrum Scale editions:

- Express Edition
- Standard Edition

Important information: Spectrum Scale Advanced Edition is not supported on the Windows operating system or with the Active File Management function.

1.6.3 IBM Spectrum Scale for Linux

Spectrum Scale for Linux systems running on IBM Power servers are supported for the following Linux distributions:

- Red Hat Enterprise Linux (RHEL) 6.2 or later.
- SUSE Linux Enterprise Server (SLES) 11 SP2 or later.
- Ubuntu 14.04.1 or later.

Spectrum Scale for Linux systems running on x86-based servers are supported for the following Linux distributions:

- Red Hat Enterprise Linux (RHEL) 6.2 or later.
- SUSE Linux Enterprise Server (SLES) 11 SP2 or later.
- Debian Linux version 6 or 7.
- Ubuntu 14.04.1 or later.

Currently, Linux operating systems support the following Spectrum Scale editions:

- Express Edition
- Standard Edition
- Advanced Edition
- File Placement Optimizer (FPO)

Linux environments also support the following functions:

- AFM
- Local read-only cache (LROC)

1.6.4 IBM Spectrum Scale for Linux on System z

Spectrum Scale 4.1 now supports Linux on System z virtual machines for the Express Edition only, which includes most base-level features. IBM intends to offer more functionality that is in the Standard and Advanced Editions in future versions of Spectrum Scale for Linux on System z.

The functions in the Express Edition include, but are not limited to the following functions:

- Snapshots
- NSD client/server capability
- Client failover
- Online or nondisruptive file system management, as adding and removing nodes and disks
- Logging

The Linux instances or nodes can be either Red Hat Enterprise Linux or SUSE Linux Enterprise Server, and they can run in LPARs or under IBM z/VM® as guest machines. The nodes also can be running on the same or different System z servers.

The Spectrum Scale for Linux on System z is supported in the following versions:

- Red Hat Enterprise Linux (RHEL) 6.5 plus service
- Red Hat Enterprise Linux (RHEL) 7.0
- SUSE Linux Enterprise Server (SLES) 11 SP3 plus service

For more information related to this subject, see the following resources:

- *Spectrum Scale for Linux on System z product* web page:

  http://ibm.co/1CEPaUq
- *Spectrum Scale for Linux on System z* in the Linux on System z Solution web page:

  http://www-03.ibm.com/systems/z/os/linux/solutions
- Spectrum Scale for Linux on System z announcement letters:

  http://www.ibm.com/common/ssi/index.wss?request_locale
- *Understanding IBM Spectrum Scale for Linux on z Systems (Express Edition)*, TIPS1211:

  http://w3.itso.ibm.com/abstracts/tips1211.html?Open

1.6.5 IBM Spectrum Scale Server solution

The IBM Spectrum Scale Server solution uses the Spectrum Scale software technology combined with dedicated storage subsystems servers. It uses the IBM Spectrum Scale Redundant Array of Independent Disks (RAID) capability to build its own cluster providing high availability and highly scalable storage resources.

The IBM Spectrum Scale RAID is a software implementation of storage RAID technologies within Spectrum Scale and it implements sophisticated data placement and error correction algorithms to deliver high levels of storage reliability, availability, and performance. Standard

Spectrum Scale file systems are created from the NSDs defined through Spectrum Scale Native RAID.

The IBM Spectrum Scale RAID integrates the functionality of an advanced storage controller into the Spectrum Scale NSD server. Unlike an external storage controller, where configuration, LUN definition, and maintenance are beyond the control of Spectrum Scale, IBM Spectrum Scale RAID itself takes on the role of controlling, managing, and maintaining physical disks, both hard disk drives (HDDs) and solid-state drives (SSDs).

Currently, the IBM Spectrum Scale RAID is available for AIX and Linux operating systems and uses the following configurations:

- IBM Power 775 Disk Enclosure.
- IBM Spectrum Scale using IBM POWER8™ technology.

When planning to use IBM Elastic Storage Server (ESS) in your current Spectrum Scale environment, the following Spectrum Scale version limitations might apply:

- IBM ESS v1.0 customers should run at GPFS v3.5.0.11 or v3.5.0.17 or higher. Specifically, v3.5.0.12 through v3.5.0.16 are unsupported.
- IBM ESS v1.5 customers should run at GPFS v3.5.0.17 or higher.
- IBM ESS v2.0 customers should run at Spectrum Scale v4.1.0.1 or higher.
- IBM ESS v2.5 customers should run at Spectrum Scale v4.1.0.5 or higher.

For more information about IBM Spectrum Scale RAID and ESS, see the following resources:

- *General Parallel File System Version 4 Release 1.0.1 Documentation Update: GSS 2.0 Information*

  http://www-01.ibm.com/support/knowledgecenter/SSFKCN/bl1du14a.pdf?lang=en
- *IBM System x GPFS Storage Server Installation, User's, and Maintenance Guide,* GA35-0001-00

  http://publib.boulder.ibm.com/epubs/pdf/a3500010.pdf
- *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00 under the *GPFS Native RAID setup and maintenance* section

  http://publibz.boulder.ibm.com/epubs/pdf/c2370320.pdf

1.7 IBM Spectrum Scale and virtualization

Spectrum Scale is supported in both dedicated or virtualized environments for AIX, Linux, or Windows operating systems platforms. Some specific requirements might apply depending on the operating system being used. In the next topics, we have more details regarding the supported virtualization features.

Note: Information about how to license GPFS in virtual environments is available at section 1.5, "IBM Spectrum Scale licensing" on page 12.

1.7.1 Virtualization on IBM Power Systems

When planning to run IBM Spectrum Scale on Power Systems, some virtualization mechanisms can be used depending on your system configuration. Currently, the following virtualization options are available on Power Systems servers:

- PowerKVM (for Linux only servers)
- PowerVM (for both Linux and AIX servers)

PowerKVM

IBM Power Systems servers have traditionally been virtualized with PowerVM, and this continues to be an option on IBM POWER8 scale-out servers.

With the introduction of the Linux only scale-out systems with POWER8 technology, a new virtualization mechanism is supported on Power Systems. This mechanism is known as a *kernel-based virtual machine* (KVM), and the port for Power Systems is called *PowerKVM*.

KVM is known as the *de facto* open source virtualization mechanism. It is used by many software companies.

IBM PowerKVM is a product that leverages the Power Systems resilience and performance with the openness of KVM, which provides several advantages:

Higher workload consolidation with processors over commitment and memory sharing:

- Dynamic addition and removal of virtual devices
- Micro threading scheduling granularity
- Integration with IBM PowerVC and OpenStack
- Simplified management using open source software
- Avoids vendor lock-in
- Uses POWER8 hardware features, such as SMT8 and micro threading

Spectrum Scale can also run on systems that are managed by PowerKVM with the considerations shown in Table 1-3.

Table 1-3 PowerKVM support matrix

Type of access	PowerKVM	Operating systems distribution	Limitations
Spectrum Scale nodes with no direct disk access	IBM_PowerKVM release 2.1.0	Linux distros supported by both PowerKVM and Spectrum Scale	► Live migration is not supported ► pKVM high availability is not supported ► Local read-only cache is not supported
Spectrum Scale nodes with direct disk access	Not supported	Not supported	Not supported

For more information about PowerKVM, see *IBM PowerKVM Configuration and Use*, SG24-8231:

http://www.redbooks.ibm.com/abstracts/sg248231.html?Open

PowerVM

When using Power Systems and AIX or Linux, we can have LPARs using physical resources, LPARs using virtual and physical resources in a mixed environment, or completely virtualized

LPARs. Spectrum Scale is supported in all configurations. This support is limited to Spectrum Scale nodes that are using the AIX V7.1 or V6.1 operating system or any supported Linux distribution.

On IBM Power Systems using PowerVM as the virtualization mechanism, it uses the following features:

- The *hypervisor* manages the memory, the processors, and the physical slots allocation.
- The *Virtual I/O Server (VIOS)* manages virtualized end devices: disks, tapes, and network sharing methods such Shared Ethernet Adapters (SEAs) and N_Port ID Virtualization (NPIV). This end-device virtualization is possible through the virtual slots that are provided, and managed by the VIOS, which can be a SCSI, Fibre Channel, and Ethernet adapter.

Virtual I/O Server (VIOS)

The VIOS is a specialized operating system that runs under a specific type of logical partition (LPAR) and it is responsible to directly access physical adapters, create virtual resources, associate physical to virtual resources, and map these virtual resources to the Virtual I/O Clients (VIOCs). The VIOC can be logical partitions running AIX or Linux (*IBM System i®* is also available but since it does not support Spectrum Scale, it will not be further mentioned in this book).

VIOS provides a robust virtual storage and virtual network solution. When two or more Virtual I/O Servers are placed on the system, it also provides high availability capabilities between them.

After the VIOS and VIOC (Client) setup is complete, the Spectrum Scale can use this virtualized infrastructure for cluster creation.

Because Spectrum Scale works by accessing the disk in parallel through multiple servers, it is required to properly set up the access control attributes, which are defined in the SAN disks that are used by Spectrum Scale, especially if using VSCSI configurations. Usually it is needed to disable two resources available in the SAN disks, the reserve_policy, and the PR_key_value. After the parameterization of the disks is defined, the disks must be allocated to the Virtual I/O Clients.

There are two VIOS methods to virtualize and enable disk access for the VIOC partitions:

- Virtual SCSI target adapters (VSCSI)
- Virtual Fibre Channel adapters (NPIV)

Virtual SCSI target adapters (VSCSI)

When using VSCSI adapter mapping (was the most common method since the earliest IBM Power5 servers), the VIOS becomes the storage provider. All disks are assigned to the VIOS, which provides the allocation to its clients through its own SCSI target initiator protocol.

The illustration for the basic configuration for a VSCSI configuration can be seen in Figure 1-2 on page 21.

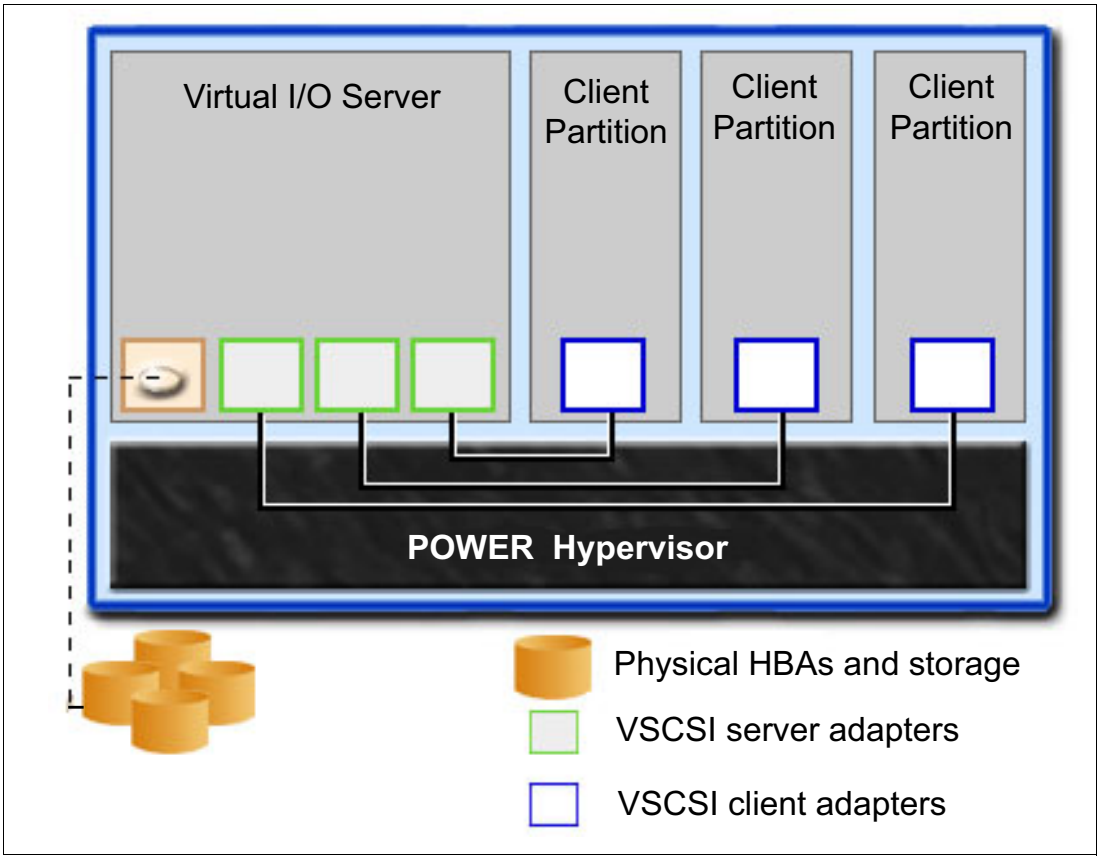


Figure 1-2 Dual VIOS VSCSI configuration example

Virtual Fibre Channel adapters (NPIV)

NPIV allows the VIOS to dynamically create multiple N_PORT IDs and share it with the VIOC.

These N_PORTs are shared through the virtual Fibre Channel adapters and provide to the VIOC a worldwide port name (WWPN), which allows a direct connection to the storage area network (SAN).

The virtual Fibre Channel adapters support connection of the same set of devices that usually are connected to the physical Fibre Channel adapters, such as the following devices:

- SAN storage devices
- SAN tape devices

Unlike VSCSI, when using NPIV the VIOS manages only the connection between the virtual Fibre Channel adapter and the physical adapter. The connection flow is managed by the hypervisor itself, resulting in a lighter implementation, requiring fewer processor resources to function when compared to VSCSI.

The NPIV functionality can be illustrated by Figure 1-3.

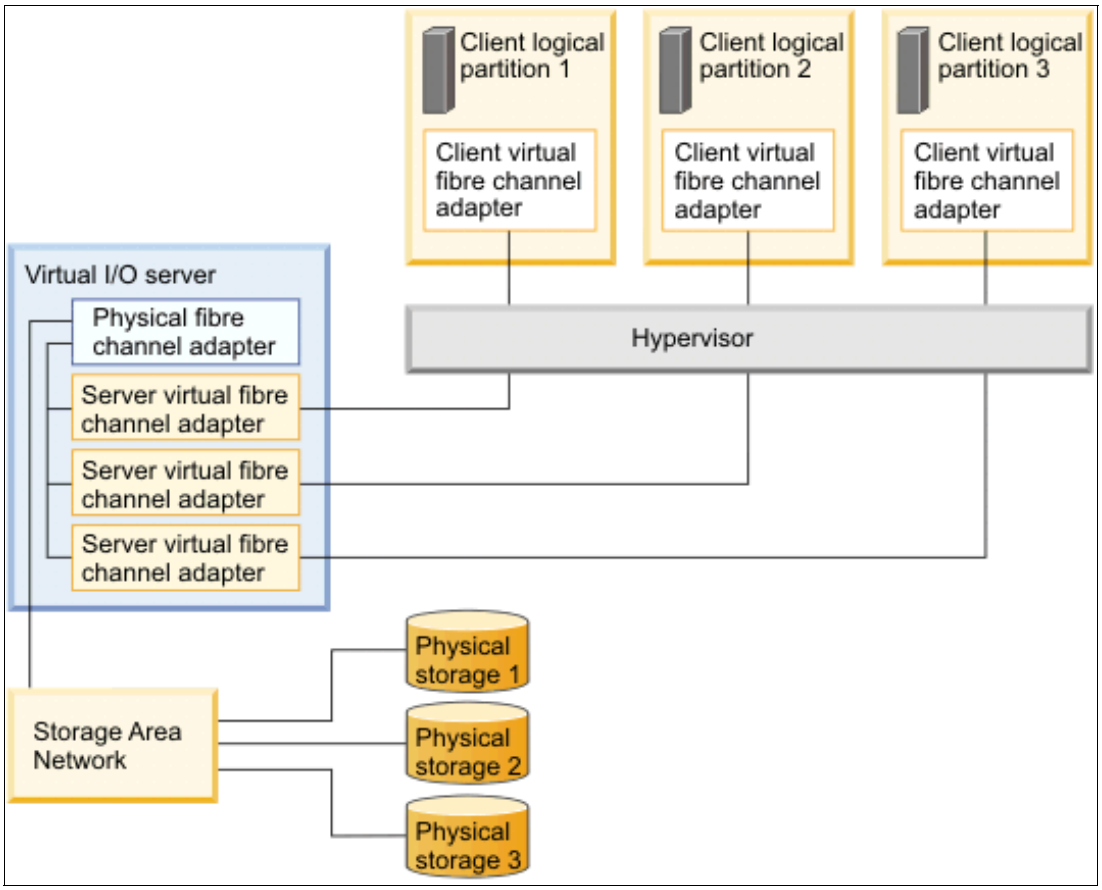


Figure 1-3 NPIV functionality illustration

Both disk allocation methods are supported for Spectrum Scale.

When planning to use Spectrum Scale in VIOS environments, check the compatibility operating system levels. The minimum required code levels for Spectrum Scale 4.1 are shown in Table 1-4.

Table 1-4 Minimum supported levels for POWER virtualization

Version	Fix level
VIOS 2.2.0.11	FP24 SP01
VIOS 2.1.1.0 or higher	
AIX 6.1 and AIX 7.1	
SLES 11	
RHEL 6 and RHEL 7	

For more information about VIOS and Virtualization on Power Systems environments, see the following sources:

- *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940

- IBM Knowledge Center:

  http://ibm.co/1CsPzYC

AIX WPAR

Another type of virtualization feature allows administrators to create *Workload Partitions* (WPARs) inside a running LPAR. The WPAR feature was implemented in AIX 6.1, and enables the administrators to virtualize the operating system without creating full LPARs on an IBM Power Systems partitioned server. WPARs provide similar levels of isolation, but without the overhead of the full system image to share common resources from the global environment. WPARs can also be used to allow previous AIX versions (5.3) to run in IBM POWER7® or IBM POWER8 hardware.

The use of WPARs to run the Spectrum Scale subsystem or mount a Spectrum Scale file system directly is not supported. Only the global environment is allowed to run the Spectrum Scale. However, a Spectrum Scale file system can be shared from the global environments to the WPAR.

To illustrate how to mount a Spectrum Scale file system in a WPAR, the following test environment was created as shown in Table 1-5.

Table 1-5 Lab system resources names for the WPAR environment

Global environment LPAR	WPAR	Spectrum Scale file system
fsaix1	gpfswpar01	/wpargpfs01

For this example, a Spectrum Scale cluster was created in the global environment partition, fsaix1, as shown in Example 1-1.

Example 1-1 The global environment running Spectrum Scale

```
GPFS cluster information
========================
  GPFS cluster name:         tsmgpfs.fsaix1
  GPFS cluster id:           12742445374757012402
  GPFS UID domain:           tsmgpfs.fsaix1
  Remote shell command:      /usr/bin/rsh
  Remote file copy command:  /usr/bin/rcp
  Repository type:           CCR

 Node  Daemon node name             IP address     Admin node name  Designation
-------------------------------------------------------------------------------
   1   fsaix1                       172.16.20.153  fsaix1           quorum-manager
   2   fsaix2                       172.16.20.154  fsaix2           quorum-manager
   3   fsaix3                       172.16.20.155  fsaix3           quorum
   4   fsaix4                       172.16.20.156  fsaix4           quorum
```

The Spectrum Scale file system is available and mounted in the Spectrum Scale server as shown in Example 1-2.

Example 1-2 Listing the Spectrum Scale file system within the global environment

```
root@fsaix1:/gpfs>mmlsmount all -L
File system wpargpvs01 is mounted on 4 nodes:
  172.16.20.153   fsaix1
  172.16.20.155   fsaix3
```

```
  172.16.20.156   fsaix4
  172.16.20.154   fsaix2

root@fsaix1:/gpfs>df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          524288    140360   74%    10140    39% /
/dev/hd2         4456448    167000   97%    42085    65% /usr
/dev/hd9var       917504    352184   62%     6444    14% /var
/dev/hd3          262144    252584    4%       56     1% /tmp
/dev/hd1          131072    130360    1%        5     1% /home
/dev/hd11admin     262144    261384    1%        5     1% /admin
/proc                  -         -    -         -     -  /proc
/dev/hd10opt      655360    372072   44%     6970    15% /opt
/dev/livedump     524288    523552    1%        4     1% /var/adm/ras/livedump
/dev/download_lv    4194304   4179472    1%        9     1% /download
/dev/fslv00       262144    186960   29%     2341    11% /wpars/GPFS_wpar
/dev/fslv01       131072    128312    3%        5     1% /wpars/GPFS_wpar/home
/opt              655360    372072   44%     6970    15% /wpars/GPFS_wpar/opt
/proc                  -         -    -         -     -  /wpars/GPFS_wpar/proc
/dev/fslv02       262144    256784    3%       23     1% /wpars/GPFS_wpar/tmp
/usr             4456448    167000   97%    42085    65% /wpars/GPFS_wpar/usr
/dev/fslv03       262144    235136   11%      382     2% /wpars/GPFS_wpar/var
/dev/wpargpvs01   20971520  20029440    5%     4038     7% /wpargpfs01
```

The WPAR gpfswpar01 has no additional file systems rather than the rootvg file systems, as shown in Example 1-3.

Example 1-3 Log in to the WPAR gpfswpar01 and listing WPAR file systems

```
root@fsaix1:/gpfs>ssh gpfswpar01
root@gpfswpar01's password:
Last login: Thu Oct  9 12:34:34 2014 on /dev/pts/0 from 172.16.20.153
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 7.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
Global            262144    186960   29%     2341    11% /
Global            131072    128312    3%        5     1% /home
Global            655360    372072   44%     6970    15% /opt
Global                 -         -    -         -     -  /proc
Global            262144    256784    3%       23     1% /tmp
Global           4456448    167000   97%    42085    65% /usr
Global            262144    235128   11%      382     2% /var
```

The Spectrum Scale file system is available in the global environment but not visible for the WPAR. Now it can be shared to the *WPAR* by using the *namefs* file system type by the following steps:

1. Ensure that you are in the global environment, create a *namefs* file system, and mount the file system as shown in Example 1-4.

Example 1-4 Creating a namefs and mounting in the global environment

```
root@fsaix1:/gpfs>crfs -v namefs -d /wpargpfs -m  /wpars/GPFS_wpar/GPFS_fs
root@fsaix1:/gpfs>mount /wpars/GPFS_wpar/GPFS_fs
```

When issuing the **crfs** command, ensure that the *-**v*** flag is set as "*namefs"*, the *-d* flag must represent the local file system in the global environment, and the ***-m*** flag must represent the mount point at the WPAR level.

2. After the file system was created and mounted within the global environment, the **df** output should list the new *namefs* called "*/wpar/GPFS_wpar/GPFS_fs"* as shown in Example 1-5.

Example 1-5 The df output should show namefs mounted

```
root@fsaix1:/gpfs>df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          524288    140352   74%    10143    39% /
/dev/hd2         4456448    167000   97%    42085    65% /usr
/dev/hd9var       917504    351656   62%     6449    14% /var
/dev/hd3          262144    252584    4%       56     1% /tmp
/dev/hd1          131072    130360    1%        5     1% /home
/dev/hd11admin     262144    261384    1%        5     1% /admin
/proc                 -         -    -         -     -  /proc
/dev/hd10opt      655360    372072   44%     6970    15% /opt
/dev/livedump     524288    523552    1%        4     1% /var/adm/ras/livedump
/dev/download_lv    4194304   4179472    1%        9     1% /download
/dev/fslv00       262144    186928   29%     2342    11% /wpars/GPFS_wpar
/dev/fslv01       131072    128312    3%        5     1% /wpars/GPFS_wpar/home
/opt              655360    372072   44%     6970    15% /wpars/GPFS_wpar/opt
/proc                 -         -    -         -     -  /wpars/GPFS_wpar/proc
/dev/fslv02       262144    256784    3%       23     1% /wpars/GPFS_wpar/tmp
/usr             4456448    167000   97%    42085    65% /wpars/GPFS_wpar/usr
/dev/fslv03       262144    235104   11%      382     2% /wpars/GPFS_wpar/var
/dev/wpargpfs01   20971520  20029440    5%     4038     7% /wpargpfs
/wpargpfs 20971520  20029440    5%     4038     7% /wpars/GPFS_wpar/GPFS_fs
```

3. After the namefs file system was created and mounted in the global environment, it is automatically displayed in the WPAR as shown in Example 1-6.

Example 1-6 df output from the WPAR

```
# df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
Global            262144    186928   29%     2342    11% /
Global            131072    128312    3%        5     1% /home
Global            655360    372072   44%     6970    15% /opt
Global                 -         -    -         -     -  /proc
Global            262144    256784    3%       23     1% /tmp
Global           4456448    167000   97%    42085    65% /usr
Global            262144    235104   11%      382     2% /var
```

```
Global          20971520  20029440    5%     4038     7% /GPFS_fs
```

When the Spectrum Scale is mounted, it can be used but not shared between other WPARs.

Because Spectrum Scale does not run in a WPAR, licenses are not required for the WPAR. Only the global instance needs to be licensed. The type of Spectrum Scale license that is required by the global instance depends on what functions the instance is performing as explained in 1.8, “Contact information” on page 28.

For more information about WPARs, see the following sources:

- IBM systems management website:

  http://www-03.ibm.com/systems/power/software/aix/sysmgmt
- *Exploiting IBM AIX Workload Partitions*, SG24-7955

1.7.2 Virtualization on x86 Linux operating systems

In a virtualization environment, the level of support depends on whether an individual Spectrum Scale node has direct-attached or SAN-attached disks. The virtualization on Linux for the x86 platform is supported when using one of the combinations as shown in Table 1-6.

Table 1-6 VM supported configurations

Type of access	KVM	OS distribution	Configuration	Limitations
Spectrum Scale nodes with no direct disk access.	RHEL 6.2-x86_64 or higher.	Linux supported by both KVM and Spectrum Scale.		► Live migration is not supported. ► KVM high availability is not supported.
Spectrum Scale nodes with direct disk access.	RHEL 6.2-x86_64 or higher.	Linux supported by both KVM and Spectrum Scale.	► PCI pass-through. ► Devices supported by Spectrum Scale and KVM. ► Virtio SCSI is supported by RHEL 6.4 or higher.	► Live migration is not supported. ► KVM high availability is not supported. ► SCSI-3 PR on Virtio is not supported.

Regarding the use of VMware for virtual Linux environments, the following combinations are observed as shown in Table 1-7.

Table 1-7 VMware supported configurations

Type of access	VMware	Operating system distribution	Configuration	Limitations
Spectrum Scale nodes with no direct disk access.	VMware ESX 4.1 or higher.	Linux supported by both VMware and Spectrum Scale.		► vShere vMotion is not supported. ► vSphere Fault Tolerance (FT) is not supported.

Type of access	VMware	Operating system distribution	Configuration	Limitations
Spectrum Scale nodes with direct disk access.	VMware ESX 5.1 or higher.	Supported only with RHEL 6.2, or higher, and SLES 11 SP2, or higher.	Pass-through Raw Device Mapping (RDM) with physical compatibility mode.	► vShere vMotion is not supported. ► vSphere FT is not supported. ► Use of Persistent Reserve is not supported. ► VMDK disks are not supported since EIO errors are not passed from ESX host to guest in APD (all paths down) condition.

Note: For details about how to configure RDM, refer to Section 14.3 *Raw Device Mapping with Virtual Machine Clusters* at:

http://vmw.re/1yw6BkP

1.7.3 Virtualization on Windows operating systems

Spectrum Scale on Windows operating systems is also supported for virtual environments. Some restrictions might be applied depending on the configuration used, specially for disk access. More information is described in Chapter 2, “Infrastructure planning and considerations” on page 29.

Regarding the use of VMware for virtual Linux environments, the following combinations are observed as shown in Table 1-8.

Table 1-8 HyperV supported configurations

Type of access	HyperV	Operating system distribution	Limitations
Spectrum Scale nodes with no direct disk access.	Windows 2008R2.	Windows 2008R2, Windows 2008R2, and Windows 7.	HyperV Live migration is not supported.
Spectrum Scale nodes with direct disk access.	None	None	None

Some additional configurations can be supported if using VMware for virtual Windows operating system environments as shown in Table 1-9 on page 28.

Table 1-9 VMware supported configurations

Type of access	VMware	Operating system distribution	Limitations
Spectrum Scale nodes with no direct disk access.	VMware ESX 5.1.	Windows 2008R2, Windows Server 2012.	► vShere vMotion is not supported. ► vSphere Fault Tolerance (FT) is not supported.
Spectrum Scale nodes with direct disk access.	None	None	None

Note: Spectrum Scale for Windows operating systems does not support any kind of raw disk I/O when running as a VM guest.

1.8 Contact information

IBM has external websites to help clients with common issues and also provides remote support for any problem regarding Spectrum Scale:

- If your question concerns a potential software error in Spectrum Scale and you have an IBM software maintenance contract, contact 1-800-IBM-SERV in the United States, or your local IBM Service Center in other countries.
- If you have a question that can benefit other Spectrum Scale users, you can post it to the Spectrum Scale technical discussion forum at:

  `http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=479&cat=13`
- For pertinent information regarding the Spectrum Scale features and common questions, check the IBM Spectrum Scale FAQ:

  `http://ibm.co/1IMGNa4`
- The FAQ is continually being enhanced. To contribute possible questions or answers, send them to:

  `mailto:gpfs@us.ibm.com`
- If you want to interact with other Spectrum Scale users, the San Diego Supercomputer Center maintains a Spectrum Scale user mailing list:

  `mailto:gpfs-general@sdsc.edu`
- You can subscribe to the list at the following location:

  `https://lists.sdsc.edu/mailman/listinfo`

2

Infrastructure planning and considerations

This chapter provides considerations for the storage and network infrastructure you can use with IBM Spectrum Scale (formerly GPFS) and also contains planning information for deploying an IBM Spectrum Scale cluster.

This chapter contains the followings topics:

- IBM Spectrum Scale cluster topologies
- Network design
- Storage design
- IBM Spectrum Scale supported platforms
- Security considerations for IBM Spectrum Scale clusters
- IBM Spectrum Scale configuration planning

© Copyright IBM Corp. 2015. All rights reserved.

2.1 IBM Spectrum Scale cluster topologies

IBM Spectrum Scale provides various configurations for customer solutions. The following four configuration types are sample IBM Spectrum Scale solutions, which are classified by the location of the application on the cluster nodes:

- Application running on Spectrum Scale NSD clients only (network-based Spectrum Scale client model).
- Application running on nodes with direct attachment to the storage (direct attached storage model).
- Application running on both NSD servers and clients (mixed NSD access model).
- Application running in an IBM Spectrum Scale File Placement Optimizer cluster (IBM Spectrum Scale FPO model).

2.1.1 Network-based Spectrum Scale client

In this configuration, the applications are running on the IBM Spectrum Scale client tier, which shares the file system access with the NSD servers using the network. The NSD servers use a direct attachment to the disk system, for example SAN or SAS.

Figure 2-1 shows the customer application running on the Spectrum Scale NSD client side. Application performance depends on the interconnect network speed.

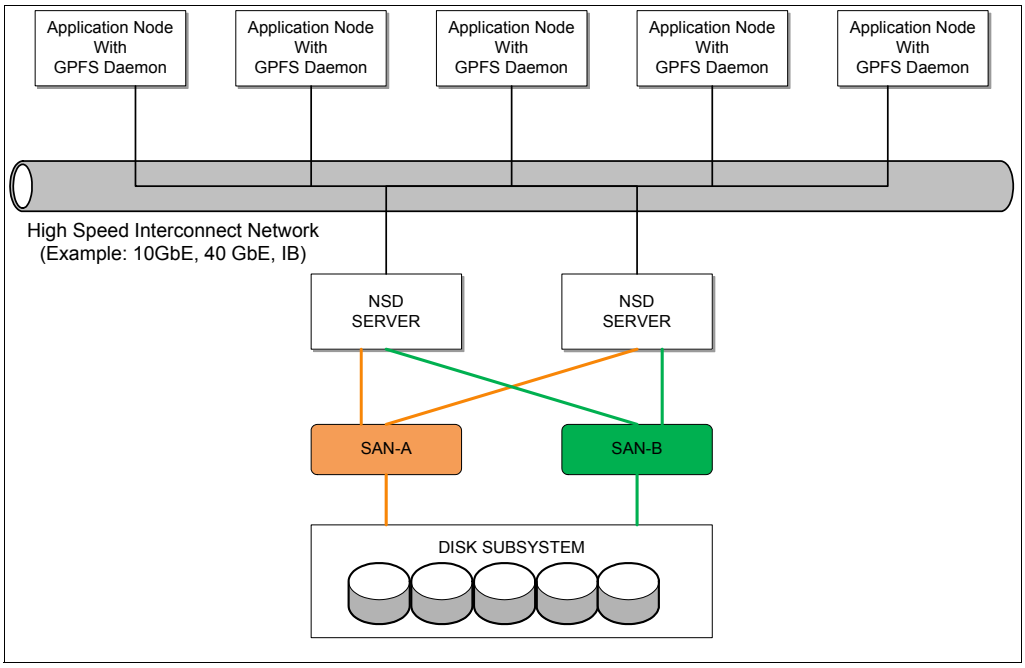


Figure 2-1 Application running on NSD clients

2.1.2 Direct attached storage

This type of configuration is used by distributed applications or workloads requiring shared storage access, for example a DB2 PureScale environment or an Oracle Real Application Cluster using IBM Spectrum Scale as a database cluster file system.

Figure 2-2 shows the application running on Spectrum Scale cluster where each node uses direct path access to the storage subsystem through the SAN. The application uses full storage performance and provides faster recovery time, when a node failure occurs, than standard shared-over-the-network NSD architecture.

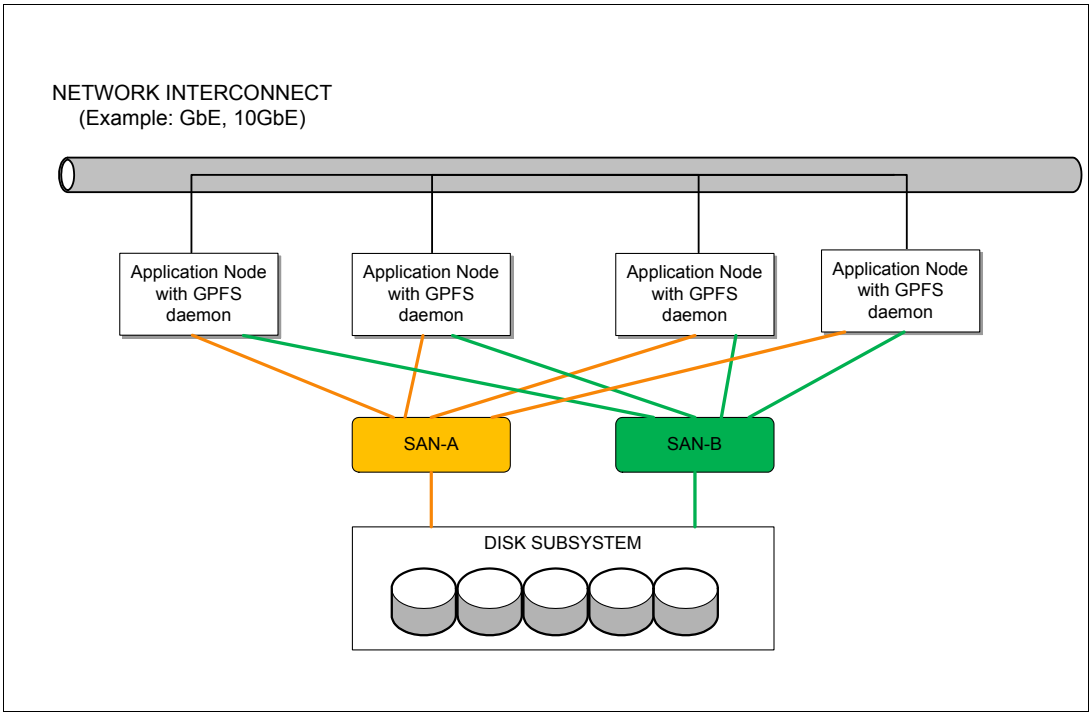


Figure 2-2 Direct storage access configuration

In a direct attached storage model, the network is used for GPFS daemon intercommunication, but all file system data is accessed directly via a storage device driver from all nodes, thus the network bandwidth requirements are lower compared with the NSD client/server model. Even if all nodes can access the disk subsystem using the SAN and defining the NSD server roles for the managed storage volumes is optional, we recommend that NSD servers still be defined. In this way, there is a failover path in case that node communication to the storage goes down.

2.1.3 Mixed NSD access: server and clients

This configuration is similar to the NSD client/server environment, except that the application components can run on both types of Spectrum Scale nodes (NSD server, NSD client). Figure 2-3 on page 32 shows a distributed application environment running on NSD clients and also on an NSD server. An example is an IBM Tivoli Storage Manager server part of a Spectrum Scale cluster, which can directly save and retrieve all data in the Spectrum Scale file systems from the cluster.

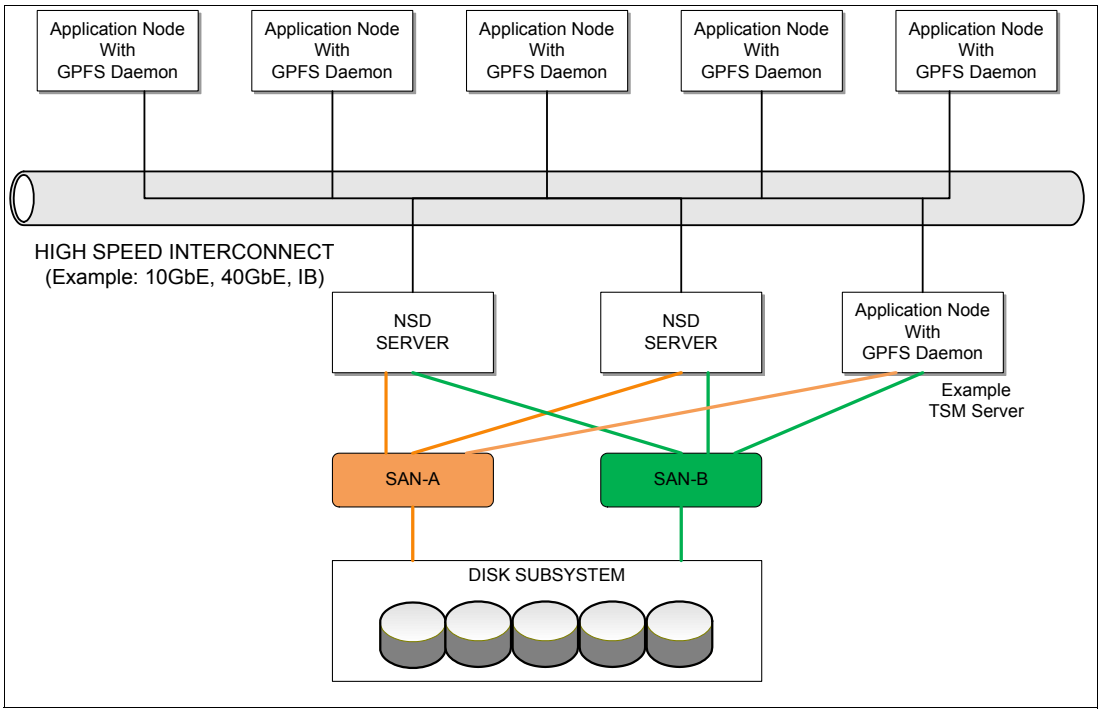


Figure 2-3 Mixed configuration with application running on NSD servers and clients

2.1.4 IBM Spectrum Scale File Placement Optimizer

This is the Spectrum Scale cluster for environments using a shared nothing architecture, where the cluster nodes have only local storage, shared with the other nodes via the interconnect network. Such examples are DB2 using the partitioning feature (DPF) or IBM InfoSphere® BigInsights™ environments, where IBM Spectrum Scale can be used as an alternative to Hadoop Distributed File System (HDFS).

Figure 2-4 shows the application running in a distributed environment, where each server uses its own (local, not shared via SAN) storage. NSD client/server communication is used to exchange data between the cluster nodes.

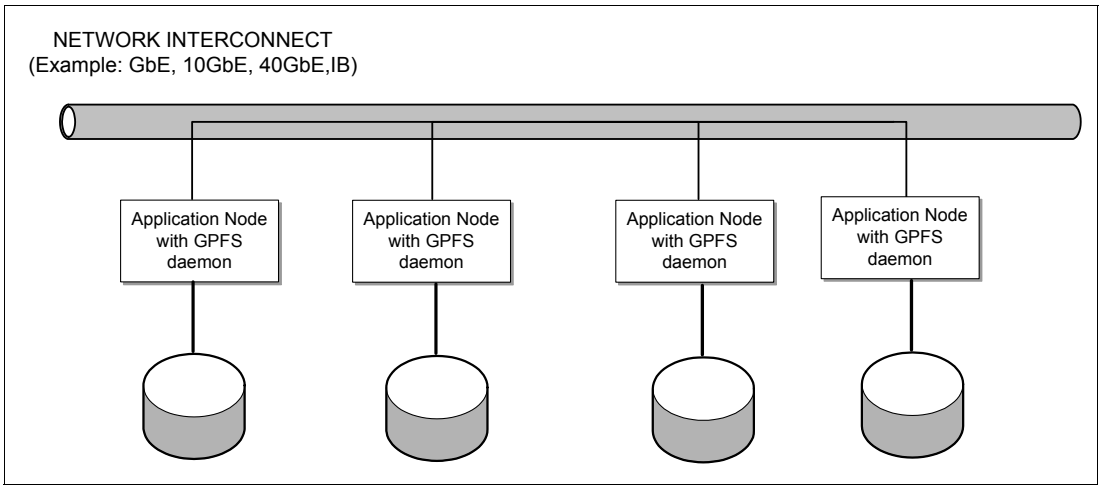

NETWORK INTERCONNECT
(Example: GbE, 10GbE, 40GbE,IB)
Application Node with GPFS daemon
Application Node with GPFS daemon
Application Node with GPFS daemon
Application Node with GPFS daemon

Figure 2-4 IBM Spectrum Scale FPO environment

Data and metadata are replicated by IBM Spectrum Scale between NSDs to ensure protection against data loss in a disk or node failure, and for maintaining access to the data in a node down event. See further details about FPO in 2.6.4, “Planning for IBM Spectrum Scale FPO” on page 93.

2.2 Network design

The IBM Spectrum Scale solutions can be deployed with various networking configurations. This section explores the major networking technologies that can be used in a Spectrum Scale cluster.

2.2.1 Ethernet adapters options

This section describes several characteristics and configuration options of the Ethernet networks that can be used when deploying the Spectrum Scale clusters.

1 Gigabit Ethernet

1 Gbps Ethernet is the most popular standard used in networking environments. Adapters can use either RJ45 or optical ports to connect to the Ethernet switches. Consider the following options regarding the Ethernet configuration with IBM Spectrum Scale:

- Jumbo frames: You can use jumbo frames to optimize the network performance for IBM Spectrum Scale, if your switching infrastructure supports it. This is a two-side configuration, which needs to be correlated: enable it on the switch side and also on the host side for the corresponding NICs. The maximum jumbo frame size differs by network interface card (NIC) adapter manufacturer. Typical value for MTU size when jumbo frame is enabled is 9000. Ensure that the MTU size is set consistent between the host NIC and the switch.
- TCP window scaling:
  - In Linux: **`/proc/sys/net/ipv4/tcp_window_scaling`** by default contains 1(enabled).
  - In AIX: Use **`ifconfig enX`** to determine the TCP windows scaling for the interfaces used by IBM Spectrum Scale. The rfc1323 can be set system wide using the **`no`** command or at the interface level using **`chdev`** or **`ifconfig`** commands.
- Tune the TCP window settings for larger values. The value of 256 KB for both TCP send and receive windows can be used as a starting point:
  - In Linux: Check `/etc/sysctl.conf`. The **`tcp_rmem`** and **`tcp_wmem`** parameters set the TCP receive and send window sizes, respectively.
  - In AIX: **`no -a`** provides a list of network options. Use **`tcp_recvspce`** and **`tcp_sendspace`** parameters to adjust the TCP window sizes. Like rfc1323, these parameters can also be set system wide using the **`no`** command, or per interface, using the **`chdev`** command.
- Configure NIC teaming: The network interface teaming is transparent to IBM Spectrum Scale. It relies on hardware and OS/drivers capabilities. Consider using teaming of Ethernet adapters for the following purposes:
  - To provide high availability for communication between the Spectrum Scale cluster nodes: If an interface fails, the traffic is directed to the available interfaces part of the teaming configuration.
  - To improve the communication bandwidth between the cluster nodes.

  See more details about Ethernet teaming in section 2.2.2, “NIC teaming configurations” on page 34.

10 Gigabit Ethernet

10 Gbps Ethernet is a technology widely adopted in many data centers. This option is useful for clustered environments with increased communication demands. 10 Gbps adapters are also available with two physical connector options: RJ45 and optical.

Even if the bandwidth might not be a concern in your environment, teaming configurations should be considered for high availability of the Spectrum Scale inter-node communication network. 10 Gbps Ethernet and higher bandwidth adapters have been proved very efficient in virtualized environments, providing a way to consolidate network traffic for multiple virtual servers.

40 Gigabit Ethernet

The 40 Gbps Ethernet technology has been recently developed and it is another option for enhancing the Spectrum Scale communication for environments with large bandwidth demands. IBM Spectrum Scale can use at this time the underlaying 40 Gbps Ethernet technology with TCP/IP communication protocol on all IBM Spectrum Scale supported platforms: AIX, Linux, and Windows operating system. The 40 Gbps adapter is available at this time for IBM Systems in several environments. The following references are some examples of available adapters at the time of writing this IBM Redbooks publication:

- IBM Flex Systems. See more details for a 40 Gbps available adapter on IBM PureFlex® systems:

  `http://www-03.ibm.com/systems/uk/flex/networking/bto/ethernet/en6132_40gb_e`

- IBM System x. See a reference for an available option for System x on:

  `http://www.redbooks.ibm.com/technotes/tips0897.pdf`

- IBM POWER8 Systems. Currently, 40 Gbps adapter is supported on POWER8 systems. See more details at the following site:

  `http://ibm.co/1xwh7xf`

For more updates, always refer to the current available list of adapters provided by the vendor for a specific system.

2.2.2 NIC teaming configurations

This section provides information about how to take advantage of features within each operating system and the network interface cards. Aggregating interfaces for use in Spectrum Scale networks provides two main benefits:

- Availability, in case of one or multiple port failures within the aggregated interface, depending on the number of participating interfaces (two or more).
- Increased bandwidth. IBM Spectrum Scale uses a TCP-based communication when using the TCP/IP communication stack (note that at this time IBM Spectrum Scale also supports RDMA in specific Ethernet configurations. See more details in section 2.2.4, “RDMA over Converged Enhanced Ethernet” on page 39). Using multiple interfaces helps when the number of nodes increases, so socket-to-socket communication between nodes gets distributed over the interfaces within the aggregates.

Bonding on Linux

The bonding module provides seven modes of operation:

- 0 for balance round robin (RR)
- 1 for active-backup
- 2 for balance-selected transmit hash policy (XOR)

- 3 for broadcast
- 4 for 802.3ad
- 5 for balance-transmit load balancing (TLB)
- 6 for balance-adaptive load balancing (ALB) change arp (CHARP)

When configuring the bonding on Linux, set it up in the network switch side also. All modules are included in all Linux distributions.

The following options can be used while configuring the network adapters:

- Round robin: The output device is selected based on the next available subordinate NIC, regardless of the source or destination of the packet.
- XOR: The **XOR** parameter selects the same subordinate for each destination hardware address.
- Active-backup: This policy ensures that one (any one) device will send and receive data at any given moment. Active-backup policy is useful for implementing high availability solutions using two network switches. Usually, 10 Gbps NIC devices are configured with failover options, and no load balancing enabled.

Teaming on Windows operating systems

The Windows operating system does not include teaming software so you must download it from Intel, Broadcom, or another vendor. The teaming function differs slightly from each network supplier, for example:

- Intel
  - Adapter Fault Tolerance (AFT)
  - Switch Fault Tolerance (SFT)
  - Adaptive Load Balancing (ALB)
  - Virtual Machine Load Balancing (VMLB)
  - Intel Link Aggregation (LA), Fast Etherchannel (FEC), and Gig Etherchannel (GEC)
  - IEEE 802.3ad
- Broadcom
  - Smart Load Balancing (SLB)
  - Link Aggregation (802.3ad)
  - Generic Link Aggregation (Trunking)
  - Failover Teaming

Etherchannel on AIX

Etherchannel is a network port aggregation technology that allows several Ethernet adapters to be put together to form a single pseudo-Ethernet device on AIX. Etherchannel has four modes:

- Standard: In this mode, the Etherchannel uses an algorithm to choose on which adapter it sends the packets out. The algorithm consists of taking a data value, dividing it by the number of adapters in the Etherchannel, and using the remainder.
- Round_robin: In this mode, the Etherchannel rotates through the adapters, giving each adapter one packet before repeating. The packets may be sent out in a slightly different order than they are given to the Etherchannel, but the Etherchannel makes the best use of its bandwidth.

- Netif_backup: This is no longer an explicit mode in the latest AIX versions (V6 and V7). It is set by choosing a primary and a backup interface. The Etherchannel activates only one interface at a time. The intention is that the Ethernet ports are plugged into separate switches, for redundancy in case of adapter or switch failure.
- 802.3ad: This mode enables the use of the IEEE 802.3ad Link Aggregation Control Protocol (LACP) for automatic link aggregation.

2.2.3 InfiniBand networks

IBM Spectrum Scale can use IP over InfiniBand and Remote Direct Memory Access (RDMA) fabric to provide access to the file system. This section describes the InfiniBand architecture.

Introducing InfiniBand

InfiniBand is an industry-standard for server I/O and inter-server communication. InfiniBand is a set of open interconnect standards and specifications developed by an industry association. Specifications and standards can be found at the following site:

http://www.infinibandta.org

InfiniBand solution provides high speed and low latency data communication and it can also simplify the integration of communications and storage networks. Usually, this architecture is used for High Performance Computing solutions because they require high speed and low latency communication. However, this technology has increased its popularity for commercial solutions, for example multimedia, storage, databases.

InfiniBand is based on a switched fabric architecture, as shown in Figure 2-5. The InfiniBand links can be connected to either host channel adapters (HCAs) that are used primarily in servers, or target channel adapters (TCAs) that are used primarily in storage subsystems.

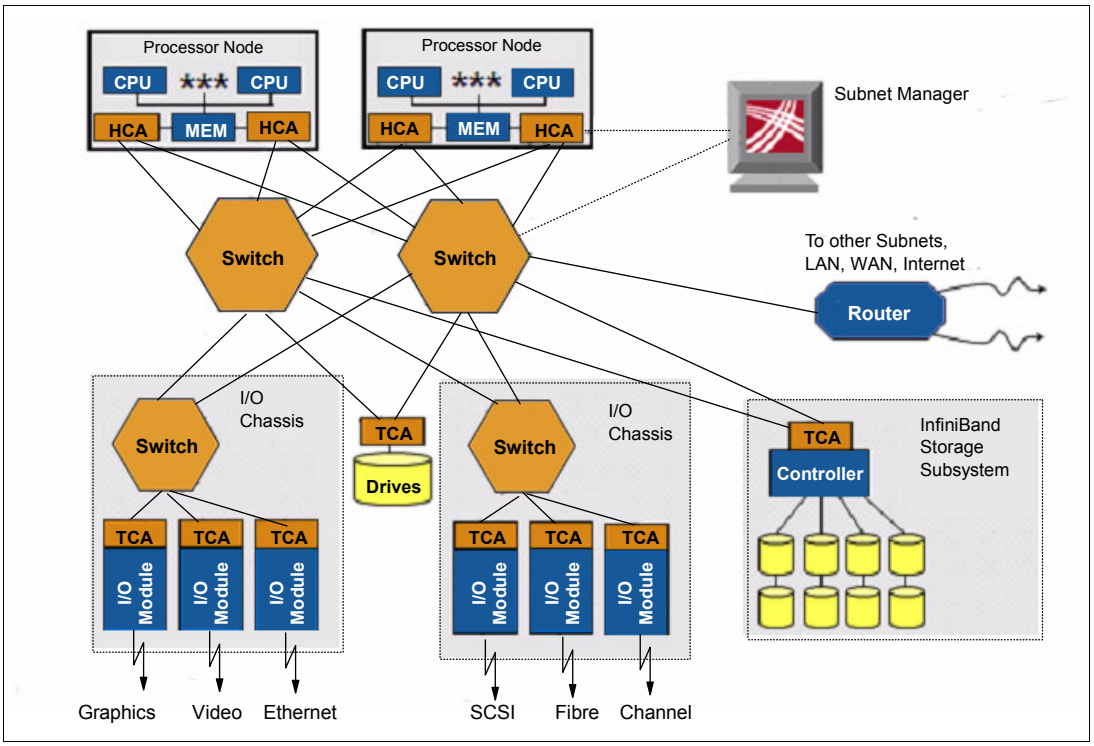


Figure 2-5 InfiniBand architecture

Overview of InfiniBand protocols

InfiniBand can be used with IBM Spectrum Scale for providing the communication between the cluster nodes, optimizing the NSD client/server communication, and also for node access to block storage attached to the InfiniBand network. Figure 2-6 shows the InfiniBand driver and the hardware stack.

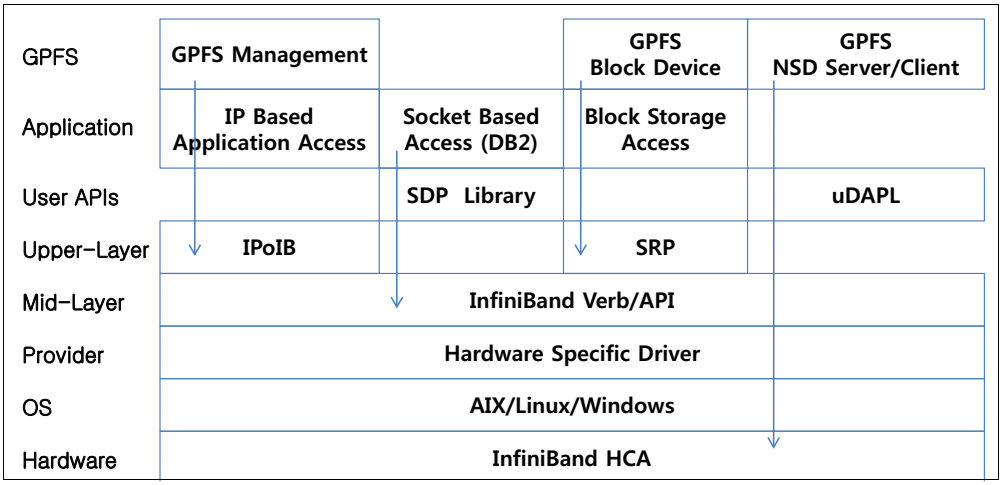


Figure 2-6 InfiniBand driver stack diagram with GPFS daemon

Upper-level protocols such as IP over InfiniBand (IPoIB), Socket Direct Protocol (SDP), SCSI RDMA Protocol (SRP), iSCSI Extensions for RDMA (iSER) and so on, facilitate standard data networking, storage, and file system applications to operate over InfiniBand. Except for IPoIB, which provides a simple encapsulation of TCP/IP data streams over InfiniBand, the other upper-level protocols transparently enable higher bandwidth, lower latency, lower CPU utilization, and end-to-end service using field-proven RDMA and hardware-based transport technologies available with InfiniBand.

IP over InfiniBand (IPoIB)

IPoIB running over high-bandwidth InfiniBand adapters can provide an instant performance boost to any IP-based applications. IPoIB supports tunneling of IP packets over InfiniBand hardware. This method of enabling IP applications over InfiniBand is effective for management, configuration, setup, and control plane-related data where bandwidth and latency are not critical. Because the application traffic runs over the standard TCP/IP networking stack, the application is completely unaware of the underlying I/O hardware.

Socket Direct Protocol (SDP)

For applications that use TCP sockets, the SDP delivers a significant performance boost and reduces CPU utilization, as well as application latency. The SDP driver provides a high-performance interface for standard socket applications and a boost in performance by bypassing the software TCP/IP stack, implementing zero copy and asynchronous I/O, and transferring data using efficient RDMA and hardware-based transport mechanisms. Applications like IBM DB2 pureScale® can use the SDP libraries and the InfiniBand infrastructure for providing a high performance and scalable communication layer between the cluster nodes.

SCSI RDMA Protocol (SRP)

SRP was defined by the ANSI TIA committee to provide block storage capable for the InfiniBand architecture. SRP is protocol that uses the InfiniBand reliable connection and RDMA capabilities to provide a high-performance transport for the SCSI protocol. SRP is similar to Fibre Channel Protocol (FCP), which uses Fibre Channel to provide SCSI over

Fibre Channel. This allows one host driver to use storage target devices from various storage hardware.

Remote Direct Memory Access (RDMA)

RDMA used to allow various servers on the InfiniBand fabric to directly access the memory of another server, eliminating the need to copy from the application memory to the operating system data buffers. As an example, a Spectrum Scale cluster or a database server cluster can use RDMA for node-to-node communication. IBM Spectrum Scale has **verbPorts** and **verbRdma** options for the InfiniBand RDMA function, and the database server cluster adds an RDMA agent to its core functionality, which allows two database instances running on different nodes to communicate directly with each other, bypassing all kernel-level communication operations.

Configuring IBM Spectrum Scale to exploit InfiniBand helps simplify the network design because InfiniBand integrates the network and the storage together (each server uses a single adapter utilizing different protocols), as in the following examples:

- Spectrum Scale cluster management network can use IPoIB.
- Spectrum Scale storage attached can use SRP.
- Spectrum Scale Network Shared Disk (NSD) communication can use RDMA.

Table 2-1 shows the platforms and protocols supported by Spectrum Scale interconnect when using an InfiniBand network.

Table 2-1 InfiniBand support for Spectrum Scale networks

Platform	Spectrum Scale network type	InfiniBand Protocol	Notes
AIX	All cluster communication	IPoIB	All supported Spectrum Scale environments.
Linux on Power/x86	admin and some daemon communication	IPoIB	All supported Spectrum Scale environments. This is a mandatory IP connectivity for both cases of NSD client/server communication using IPoB or RDMA.
	NSD client/server	IPoIB	All supported Spectrum Scale environments.
		RDMA	Requirements: ► Supported on Linux x86_64 and ppc64. ► RDMA over IBM supported on the following RDMA stacks: Linux Distro RDMA, Mellanox OFED 1.5.x and 2.x, OpenFabrics OFED 1.5.x and OFED 2.x. ► Mellanox HCAs. ► RDMA connection manager feature requires all nodes have a minimum version of GPFS V3.5.0.11 or later Spectrum Scale versions.
Windows operating system	All cluster communication	IPoIB	All supported Spectrum Scale environments.

Refer to IBM Spectrum Scale Frequently Asked Questions for any updates regarding InfiniBand support with IBM Spectrum Scale:

http://ibm.co/1IKO6PN

2.2.4 RDMA over Converged Enhanced Ethernet

RDMA over Converged Enhanced Ethernet (RoCEE) is a link layer protocol that provides efficient low-latency RDMA services over Layer 2 Ethernet. For various reasons, Ethernet is often the choice in data centers and hence, it is essential to bring the advantages RDMA provides to IBM Spectrum Scale into the Ethernet environment as well.

We currently support RDMA over InfiniBand (IB). With IBM Spectrum Scale 4.1.0.4, we do support RDMA over Ethernet networks by encapsulating IB style headers in Ethernet packets. This line item extends the current IBM Spectrum Scale support for using RDMA over IB to RoCEE environment. Except for a few configuration settings and tuning, IBM Spectrum Scale works in RoCEE-based systems nearly similar. To the user, it does not matter whether the underlying network is an IB network or Enhanced Ethernet.

At the time of writing, the Mellanox OFED driver is mandatory for running RDMA over Ethernet. RoCEE is supported on Linux only.

In the following sections, we explain how to set up RoCEE for usage with IBM Spectrum Scale.

Network considerations

In general, any kind of Ethernet network can be used for configuring a RoCEE capable environment. There is no dedicated recommendation concerning the network bandwidth. There is no need to have a 40 GB Ethernet available. RoCEE could run fine in a 10 GB as well and theoretical in a single GB network, too.

But you do have to fulfill some network characteristics inside your Ethernet switch configuration. Because we intend to access memory of foreign members in the network directly, the Ethernet packets from our network traffic should be handled differently, with a higher priority. Therefore, the network should be configured as a lossless configuration with flow control disabled. Verify the network environment with your network administrator. See the Mellanox RoCEE website:

http://www.mellanox.com/page/products_dyn?product_family=79&mtag=roce

The preceding website links to an application note on running RoCEE over L2 networks enabled with PFC, and a white paper on RoCEE in the data center. Priority Frame Control (PFC) has to be configured within the operating system and within the network/switch environment as well. The result is a so called *lossless transport layer*, which we need for RDMA over converged enhanced Ethernet.

Installing OFED driver

If you want to use RDMA over your Ethernet, every node in your cluster needs the appropriate OFED driver installed. So RoCEE support for IBM Spectrum Scale is based on the Open Fabrics Enterprise Distribution (OFED) driver. While it is common that most of enterprise Linux OSs already provide its own package with its distribution, we recommend using the Mellanox OFED packages with a minimum release level from at least 2.1 or higher. Check for new driver updates for OFED on Mellanox website:

http://www.mellanox.com

You find the OFED driver package under → **Products** → **Accelerator Software** → **Linux SW/Drivers**.

The OFED manuals are at the following website:

http://bit.ly/1EZXmvw

In addition, depending on the used adapter and operating system, you should download your latest firmware and driver image according to your release level of your operating system. Unpack the tar file as shown in Example 2-1.

Example 2-1 Downloading OFED

```
[root@gss01 mnt]# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 6.4 (Santiago)

[root@gss01 mnt]# ls
MLNX_OFED_LINUX-2.1-1.0.0-rhel6.4-x86_64.tar
[...]
# untar it and step into the directory
[root@gss01 MLNX_OFED_LINUX-2.1-1.0.6-rhel6.4-x86_64]#
```

Start the installation procedure as shown in Example 2-2.

Example 2-2 Installing OFED

```
[root@gss01 MLNX_OFED_LINUX-2.1-1.0.6-rhel6.4-x86_64]# ./mlnxofedinstall
Logs dir: /tmp/MLNX_OFED_LINUX-2.1-1.0.6.32531.logs
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Do you want to continue?[y/N]:y

Uninstalling the previous version of MLNX_OFED_LINUX

Starting MLNX_OFED_LINUX-2.1-1.0.6 installation ...

Installing mlnx-ofa_kernel RPM
Preparing...                ##################################################
mlnx-ofa_kernel             ##################################################
[...]
INFO: updating ...

IMPORTANT NOTE:
===============

- The FCA Manager and FCA MPI Runtime library are installed in /opt/mellanox/fca directory.
- The FCA Manager will not be started automatically.
- To start FCA Manager now, type:
    /etc/init.d/fca_managerd start

- There should be single process of FCA Manager running per fabric.

- To start FCA Manager automatically after boot, type:
    /etc/init.d/fca_managerd install_service

- Check /opt/mellanox/fca/share/doc/fca/README.txt for quick start instructions.

Preparing...                ##################################################
dapl                        ##################################################
```

```
Preparing...                ##################################################
[...]
- Changing max locked memory to unlimited (in /etc/security/limits.conf)
  Please log out from the shell and login again in order to update this change
  Read more about this topic in the VMA's User Manual

- VMA README.txt is installed at: /usr/share/doc/libvma-6.5.9-0/README.txt
- Please refer to VMA journal for the latest changes: /usr/share/doc/libvma-6.5.9-0/journal.txt
Preparing...                ##################################################
mlnxofed-docs               ##################################################
Preparing...                ##################################################
mpitests_mvapich2__1_9      ##################################################
Preparing...                ##################################################
mpitests_openmpi__1_6_5     ##################################################
Preparing...                ##################################################
mpitests_openmpi__1_7_4     ##################################################
Device (1b:00.0):

[...]
Installation finished successfully.

[root@gss01 MLNX_OFED_LINUX-2.1-1.0.6-rhel6.4-x86_64]#
```

After the installation, reboot the machine. After the reboot, OFED is ready for use.

Configuring adapter and network for RoCEE

At the time of writing, only *one* Ethernet Port for RoCEE is supported for usage with IBM Spectrum Scale based on the OFED driver. However, the current released OFED v2.3 supports to run more than one adapter within the same subnet and doing RoCEE.

A working RoCEE communication relies on Ethernet, and so every interface card has to get a valid IP address. Every node within the Spectrum Scale cluster should be able to communicate to each other as usual by this IP. You can configure your Spectrum Scale cluster with an appropriate subnet parameter to use the network for TCP/IP daemon communication as well, but it is not a must have. The TCP/IP Spectrum Scale daemon communication can be configured on another network as well.

Configure your interfaces concerning your operating system guidelines. Bonding for RoCEE is not advised. If you start using your MLX connect-X cards for the first time, you might need to adjust the transport setting of the adapter by the **connectx_port_config** command to create a network interface inside your operating system.

The **connectx_port_config** command comes with the OFED driver as shown in Example 2-3.

Example 2-3 Output of connectx_port_config

```
[root@gss01 network-scripts]# connectx_port_config

ConnectX PCI devices :
|---------------------------|
| 1             0000:1b:00.0 |
| 2             0000:86:00.0 |
|---------------------------|

Please select device to modify [1]: 2
0000:86:00.0

Before port change:
auto (ib)
```

```
auto (ib)

|---------------------------|
| Possible port modes:      |
| 1: Infiniband             |
| 2: Ethernet               |
| 3: AutoSense              |
|---------------------------|
Select mode for port 1 (1,2,3): 2
Select mode for port 2 (1,2,3): 2

After port change:
eth
eth
[root@gss01 network-scripts]#
```

Use the **ibdev2netdev** command to verify your interface-names as shown in Example 2-4.

Example 2-4 Output of ibdev2netdev

```
[root@gss01 network-scripts]# ibdev2netdev
mlx4_0 port 1 ==> eth2 (Down)
mlx4_0 port 2 ==> eth3 (Down)
mlx4_1 port 1 ==> eth4 (Down)
mlx4_1 port 2 ==> eth5 (Down)
```

Configure the ethX devices as usual with a valid IP address. We create a small example. We take two nodes out of an existing cluster and show how to configure RoCEE. See Figure 2-7.

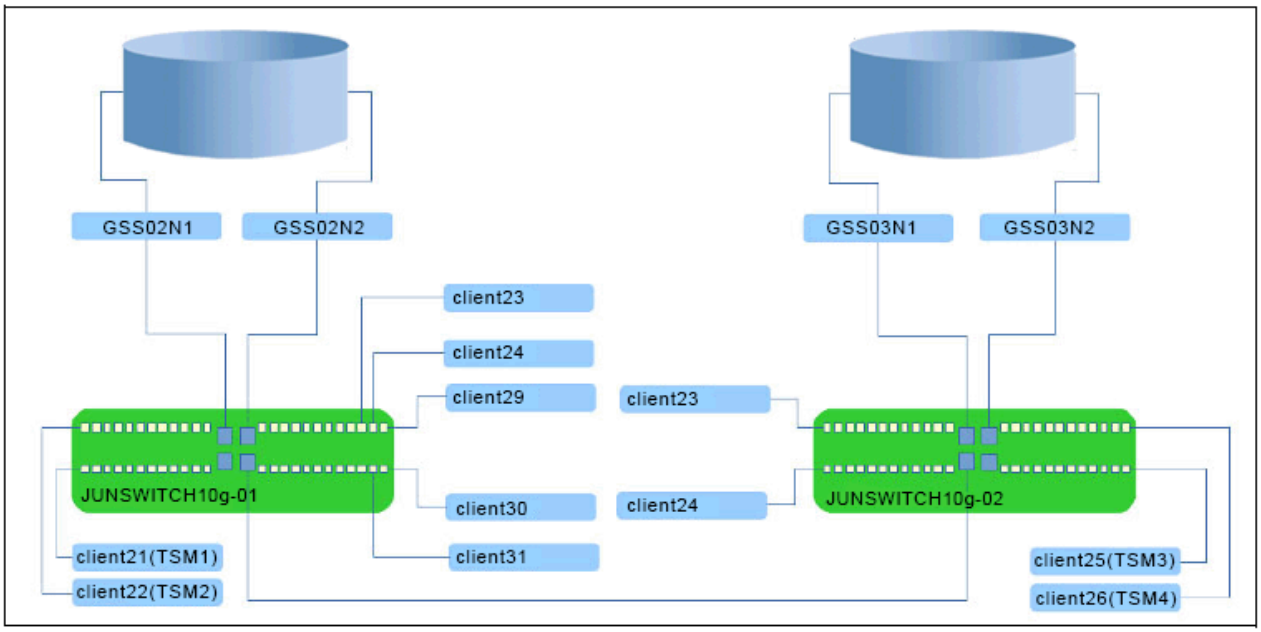


Figure 2-7 Sample RoCEE connectivity environment

The cluster configuration is shown in Example 2-5.

Example 2-5 Sample cluster connectivity and configuration

```
[root@client21 ~]# mmlscluster
[....]
Node  Daemon node name                IP address     Admin node name                 Designation
-----------------------------------------------------------------------------------------------
   1   gss02n1.sonasad.almaden.ibm.com   192.167.6.1    gss02n1.sonasad.almaden.ibm.com   quorum-manager
   2   gss03n1.sonasad.almaden.ibm.com   192.167.16.1   gss03n1.sonasad.almaden.ibm.com   quorum-manager
   3   client21.sonasad.almaden.ibm.com  192.167.13.21  client21.sonasad.almaden.ibm.com  quorum-manager
   4   client25.sonasad.almaden.ibm.com  192.167.13.25  client25.sonasad.almaden.ibm.com  quorum-manager
   5   client26.sonasad.almaden.ibm.com  192.167.13.26  client26.sonasad.almaden.ibm.com  quorum-manager
   6   gss02n2.sonasad.almaden.ibm.com   192.167.6.2    gss02n2.sonasad.almaden.ibm.com   manager
   7   gss03n2.sonasad.almaden.ibm.com   192.167.16.2   gss03n2.sonasad.almaden.ibm.com   manager
   8   client22.sonasad.almaden.ibm.com  192.167.13.22  client22.sonasad.almaden.ibm.com  manager
```

The first step is to verify the TCP/IP connectivity and when it works, you can test RDMA as well. Refer to Example 2-6.

Example 2-6 Usual ping test

```
[root@client21 ~]# ping client25
PING client25.sonasad.almaden.ibm.com (192.167.13.25) 56(84) bytes of data.
64 bytes from client25.sonasad.almaden.ibm.com (192.167.13.25): icmp_seq=1 ttl=64 time=1.55 ms
[...]

[root@client21 ~]# ping client26
PING client26.sonasad.almaden.ibm.com (192.167.13.26) 56(84) bytes of data.
64 bytes from client26.sonasad.almaden.ibm.com (192.167.13.26): icmp_seq=1 ttl=64 time=0.706 ms
root@client21 ~]#
```

In the next examples, we demonstrate how to test RDMA over Ethernet. For doing so, we use the **ib_read_bw** command, which comes with the Mellanox OFED driver. The behavior of this tool is similar to **iperf**. Start it in server mode (default) on one node, and initiate a test connection by calling it from a remote node as shown in Example 2-7.

Example 2-7 Starting RoCEE test on node

```
[root@client21 ~]# ib_read_bw

************************************
* Waiting for client to connect... *
************************************

[...] nothing happens .. it waits for the connect from remote...
```

Keep the window open and open a second window on any other node in your cluster. Now, initiate the command on one (other) node in the cluster to see if RoCEE works as shown in Example 2-8 on page 44. When running the command on the *client* side, you have to specify manually where to connect. So we choose in our example to connect from client25 to client21. In both windows, you receive a summary message as shown in Example 2-8 on page 44.

Example 2-8 RoCEE test and results

```
root@client25 ~]# ib_read_bw client21

---------------------------------------------------------------------------------------
                    RDMA_Read BW Test
 Dual-port       : OFF          Device         : mlx4_0
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 TX depth        : 128
 CQ Moderation   : 100
 Mtu             : 1024[B]
 Link type       : Ethernet
 Gid index       : 0
 Outstand reads  : 16
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x027d PSN 0x148fae OUT 0x10 RKey 0x8013703 VAddr 0x007fece3f90000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:255:15
 remote address: LID 0000 QPN 0x0277 PSN 0x407132 OUT 0x10 RKey 0x8013703 VAddr 0x007fea0b490000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:192:168:255:14
---------------------------------------------------------------------------------------
 #bytes     #iterations    BW peak[MB/sec]    BW average[MB/sec]   MsgRate[Mpps]
 65536      1000             1089.70            1089.70            0.017435
---------------------------------------------------------------------------------------
[root@client25 ~]#
```

When you get similar results with your RoCEE test, proceed to configure IBM Spectrum Scale for using RoCEE.

Cluster configuration for RoCEE

We now configure IBM Spectrum Scale to use the already prepared network interfaces. First, configure IBM Spectrum Scale to map which network adapter cards and ports should be used with RoCEE. Verify the output from the **ibdev2netdev** command as shown in Example 2-4 on page 42 again, and configure your cluster settings accordingly. In addition, enable IBM Spectrum Scale for RDMA communication with the two verbs parameter as shown in Example 2-9. The verbsPorts setting might be different within your nodes, thus this parameter can be adjusted on a per node base.

Example 2-9 IBM Spectrum Scale settings

```
[root@client21 ~]# ibdev2netdev
mlx4_0 port 1 ==> eth0 (Up)
mlx4_0 port 2 ==> eth1 (Down)
[root@client21 ~]#

[root@client21 ~]# mmlsconfig
[...]
verbsPorts mlx4_0/1/0
verbsRdma enable
verbsRdmaCm enable
[...]
```

Restart your Spectrum Scale daemon and monitor the `mmfs.log` file. You should see a message similar to Example 2-10.

Example 2-10 mmfs.log.latest when starting verbs RDMA

```
Sat Nov 15 08:33:49.895 2014: [I] VERBS RDMA starting.
Sat Nov 15 08:33:49.896 2014: [I] VERBS RDMA library librdmacm.so (version >= 1.1) loaded and initialized.
Sat Nov 15 08:33:53.103 2014: [I] VERBS RDMA device mlx4_0 port 1 fabnum 0 opened, lid 0, 1x QDR ETHERNET.
Sat Nov 15 08:33:53.104 2014: [I] VERBS RDMA mlx4_0 port
 1 using network interface eth0, IP address 192.168.255.14
Sat Nov 15 08:33:53.105 2014: [I] VERBS RDMA started.
```

Monitoring RoCEE

To verify all the RDMA connections in your cluster, use the **mmdiag** command. See Example 2-11.

Example 2-11 Monitoring RDMA connections

```
[root@client21 ~]# mmdiag --network

=== mmdiag: network ===

Pending messages:
  (none)
[....] some output is depressed due to better overview ... [...]

RDMA Connections between nodes:
  Fabric 0 - Device mlx4_0 Port 1 Width 1x Speed QDR lid 0
    hostname                        idx CM state VS buff RDMA_CT(ERR) RDMA_RCV_MB RDMA_SND_MB VS_CT(ERR) VS_SND_MB VS_RCV_MB WAIT_CON_SLOT WAIT_NODE_SLOT
    client25                        0   Y  RTS   (Y)1536 0      (0  ) 0           0           2280 (0  ) 0         0         0             0
    client26                        0   Y  RTS   (Y)1536 0      (0  ) 0           0           3447 (0  ) 0         0         0             0
    client22                        0   Y  RTS   (Y)1536 0      (0  ) 0           0           2283 (0  ) 0         0         0             0
[root@client21 ~]#
```

2.2.5 IBM data center networking products

The IBM data center networking (DCN) products offering provides the entire network solution and the network management infrastructure. When you design a large cluster with IBM Spectrum Scale, consider as a design point the network and how to manage the cluster within the network. The following items are part of the data center networking product solution:

- 1 Gbps Switch, 10 Gbps Switch, 40 Gbps Switch
- 10 Gbps FCoE Switch
- Backbone switch
- Network management software

Note: See the following resource for more information:

http://www.ibm.com/systems/networking

2.3 Storage design

There are several aspects to consider when designing the storage configuration for a Spectrum Scale cluster. You need to select the appropriate storage solution to help you achieve the wanted performance for your Spectrum Scale cluster.

The following list presents several storage technologies that you might consider when designing the storage solution for your Spectrum Scale implementation:

- Server connectivity:
  - Fibre Channel host bus adapter (HBA)
  - SAS HBA
  - IB HBA (SDR)
  - FCoE
- Fabric infrastructure: SAN, SAS, IB
- Storage subsystem connectivity, backend DASD (FC, SATA, SAS, SSD), and the RAID type

The following sections detail the HBA characteristics, the multipath options, and also the storage considerations for Spectrum Scale clusters.

2.3.1 Host bus adapter

This section provides details about the characteristics of the HBA.

The host bus adapter characteristics

The HBA is most often used to refer to a Fibre Channel interface. The two types of worldwide names (WWNs) on a host bus adapter are as follows:

- A worldwide node name (WWNN), which is shared by all ports on a host bus adapter.
- A worldwide port name (WWPN), which is unique to each port.

HBA models have various speeds: 1 Gbps, 2 Gbps, 4 Gbps, 8 Gbps, or 16 Gbps.

For IBM Spectrum Scale, we recommend using the latest releases of firmware and HBA driver.

N_Port ID Virtualization (NPIV) architecture

Figure 2-8 shows how to assign a volume on each virtual machine or logical partition. The layers for NPIV configuration differ on System x and Power Systems.

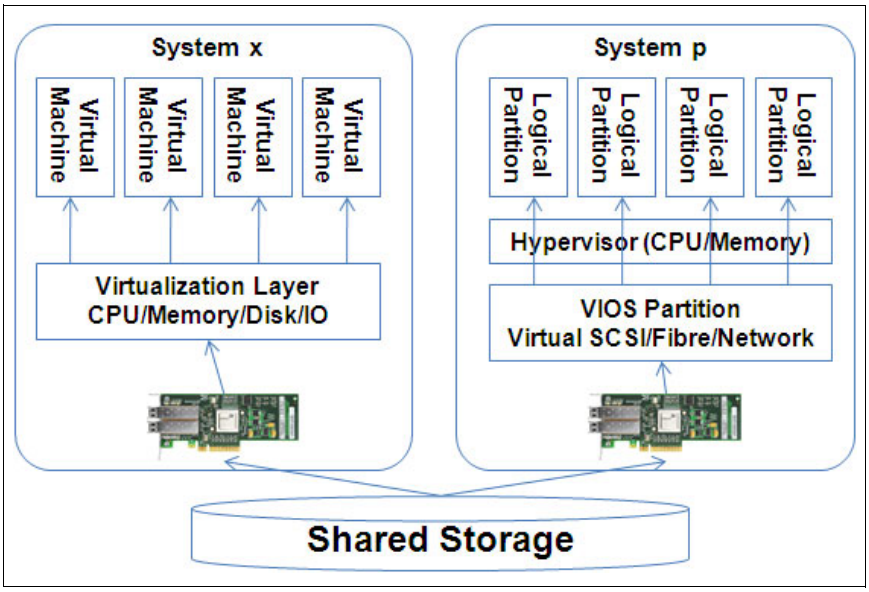


Figure 2-8 Managed system configured to use NPIV on System x and IBM System p®

NPIV technology was introduced with System z in 2007. NPIV provides a simple solution for physical HBA sharing, with good availability capabilities. NPIV uses HBA technology, which creates virtual HBA ports on a host by abstracting the underlying physical port. This support enables a single physical Fibre Channel HBA port to function as multiple logical ports, each with its own (WWPN) identity.

NPIV is an ANSI standard that describes how a single Fibre Channel port can register with several WWPNs or multiple N_Port IDs (also called FCIDs) in the SAN fabric. NPIV devices, which are connected to the same switch port, have a unique device WWPN and a unique 24-bit address (FCID).

A particular case in handling of the NPIV WWPNs applies for Power Systems LPAR using Live Partition Mobility (LPM). The virtualization I/O layer (VIOS) provides two WWPNs for each virtual Fibre Channel port (a primary/active and a secondary/standby), but only a primary WWPN is used in normal operation. The secondary WWPN is used during the LPM process. For more information, see the LPM web page in the IBM Knowledge Center:

http://ibm.co/1MvoAn6

Considerations for using NPIV:

- Using NPIV, a single physical server connection is able to provide independent storage access to multiple virtual servers.
- A virtual server requires secure access to storage in the same way as a physical server does. Using NPIV, you have the flexibility share or not the access between the virtual servers to the storage resources in SAN. Security access rules can be defined with logical zoning (based on WWPN addresses).

2.3.2 Multipath driver

The following section provides characteristics and details of the multipath driver on Power Systems and System x. For specific storage and supported multipath configurations with IBM Spectrum Scale, see the IBM Spectrum Scale FAQs document:

http://ibm.co/1IKO6PN

IBM Spectrum Scale automatically detects the multipath devices used on different platforms (AIX, Linux, Windows operating system) in supported configurations. If some exceptions apply in naming of the multipath devices, IBM Spectrum Scale can call a user-provided script to map the names of the multipath devices to a Spectrum Scale known type. For a list of Spectrum Scale known devices, see 2.6.2, “Network Shared Disk creation considerations” on page 78.

Multipath driver MPIO on AIX

Multiple Path I/O (MPIO) provides increased availability of external disk resources by providing redundant paths to the resource. The MPIO driver is natively supported in AIX and can operate in two modes:

- Active-passive (failover): Only one path to a device is active for the I/O transfer.
- Active-active (round robin): Multiple paths to a device are concurrently used for the I/O transfer. In some configurations, MPIO can detect if the load on some paths increases and prioritizes the traffic to other available paths.

There are several MPIO drivers on AIX:

- Native MPIO driver with supported storage devices from IBM or other vendors.

- IBM Subsystem Device Driver Path Control Module (SDDPCM), which runs on top of the native MPIO driver, provides additional capabilities. This driver is used with most of the current storage devices from IBM: Storwize® V7000, SAN Volume Controller, or DS8000. For more details about SDDPCM for AIX, see the following site:

  http://www-01.ibm.com/support/docview.wss?uid=ssg1S4000203

- OEM multipath drivers, such as Hitachi Dynamic Link Manager (HDLM) or EMC Powerpath supporting AIX MPIO, for OEM storage that is attached to AIX systems.

Depending on the vendor and the storage device, there might also be device drivers that are not based on MPIO. IBM provides SDD for AIX for IBM storage, which is a pseudo device driver designed to support multipath functions. This device driver creates special multipath devices (called vpath) based on regular hdisk devices. Their use is deprecated at this time, and the customer should consider using MPIO. You can see more details about SDD for AIX at the following site:

http://www-01.ibm.com/support/docview.wss?uid=ssg1S4000065

In a PowerVM virtual environment, when using storage resources provided through the Virtual I/O Server (VIOS), consider using two VIOS per Power system for redundancy. You have the following options for accessing the external storage resources:

- Virtual SCSI: LUNs from storage are mapped to the VIOS and mapped from VIOS to the client partitions.
- NPIV: Each client LPAR has its own virtual Fibre Channel adapter, while the VIOS owns the physical FC HBAs configured in pass-through mode.

Figure 2-9 illustrates a comparison between the two attachments for external storage to client LPARs, and distributed in different frames. You can observe in Figure 2-9 that NPIV provides the mechanism for the partition to have direct access to the SAN resources, while using the vSCSI protocol to map the external disks to VIOS and then define additional vSCSI mapping to the client partition.

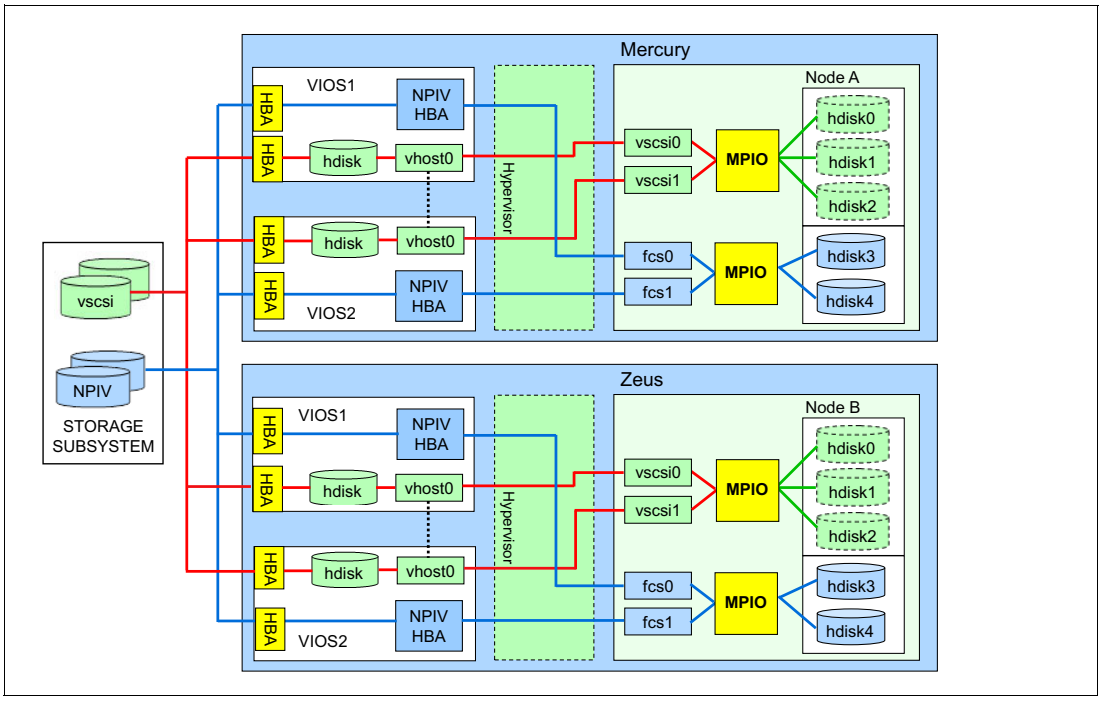


Figure 2-9 Multipath access to external storage resources using MPIO

Note: IBM Spectrum Scale supports VSCSI and NPIV virtualization technologies provided by PowerVM on AIX and Linux on POWER LPARs. See more details about the supported combinations of the operating system, VIOS, and IBM Spectrum Scale versions on IBM Spectrum Scale FAQs:

http://ibm.co/1IKO6PN

Multipath driver on Linux

The native Device Mapper (DM) multipath driver for Linux, which is provided by the open source community, is the solution for multipath implemented for most storage configurations with Linux. The solution provides the following elements:

- Failover using an active-passive configuration: In this type of configuration, only half the paths are used at any time for I/O. If any element of an I/O path (the cable, switch, or controller) fails, DM multipath switches to an alternate path.
- Load sharing using the active-active mode: I/O is spread over the paths in a round-robin fashion. In certain configurations, the DM-multipath can detect loading on the I/O paths and dynamically rebalances the load. With DM multipath configured, a failure in these points causes the DM-multipath to switch to the alternate I/O path.

You can read more details about the device mapper driver at the following website:

http://red.ht/19GnWRE

Multipath driver on Windows operating system

The Windows operating system MPIO is the framework provided by Microsoft to allow storage providers to develop multipath solutions for their specific hardware. These modules provided by the vendor are called device-specific modules (DSMs). MPIO can be used with Fibre Channel, iSCSI, or SAS interfaces to provide multipath functionality in Windows Server 2008, Windows Server 2008 R2, and Windows Server 2012. For more information about MPIO support for Windows operating systems, see the following site:

http://technet.microsoft.com/en-us/library/ee619734%28v=ws.10%29.aspx

For IBM storage devices, the specific DSM is included in the Subsystem Device Driver Device Specific Module (SDDDSM) package. See more details about SDDDSM and supported devices on the following site:

http://www-01.ibm.com/support/docview.wss?uid=ssg1S4000350

Multipath driver with the IBM SAN Volume Controller

The IBM System Storage SAN Volume Controller provides virtualized volume with heterogeneous storage. For example, a single service point of view can be configured with the multipath driver provided by the storage vendor. If you want to configure a multipath environment with multiple vendor storage controllers, the SAN Volume Controller multipath driver can be installed in your system because separate drivers might not be installed in one system. SAN Volume Controller creates a virtualization layer for all underlying storage subsystems and provides uniform access for a heterogeneous environment using a single multipath driver.

Figure 2-10 shows the SAN Volume Controller (SVC) architecture and the physical and logical views.

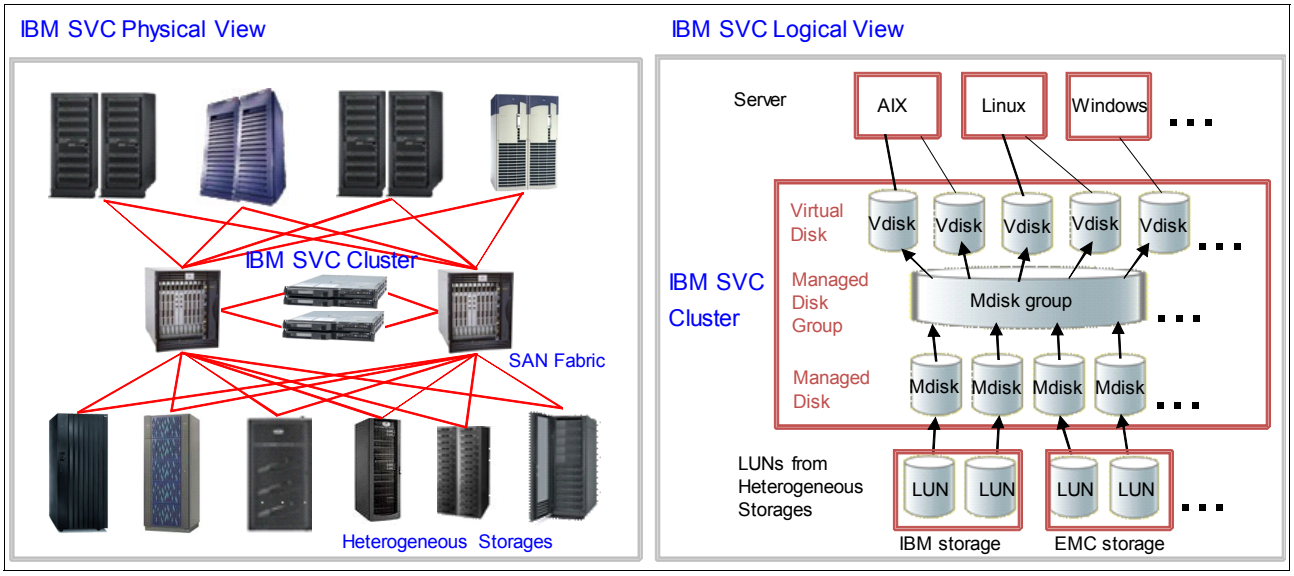


Figure 2-10 IBM SAN Volume Controller physical and logical view

The hosts must run a multipath device driver to show a single device. The multipath driver supported and delivered by SAN Volume Controller is the IBM Subsystem Device Driver, which provides two options: SDDPCM (based on MPIO) or SDD. Native MPIO drivers on selected hosts are also supported.

Notes: For operating system-specific information about MPIO support, refer to the following website:

`http://www.ibm.com/systems/storage/software/virtualization/svc/interop.html`

The support matrix for the Subsystem Device Driver on different platforms can be read at the following website:

`http://www-01.ibm.com/support/docview.wss?uid=ssg1S7001350`

2.3.3 Storage subsystem considerations

This section presents considerations for storage planning for your Spectrum Scale cluster.

RAID level

Choose the appropriate RAID protection level for your array. For most situations, you need a degree of availability protection, which cannot be provided with RAID 0. You also have to select the best RAID level according to your applications and business needs. Use Table 2-2 as a guideline for selecting RAID protection for IBM Spectrum Scale. In the first four lines of the table, we use a number between one (first option) and three (last option) for the specified criteria in the first column of the table, for the most used RAID level configurations.

Table 2-2 Guidelines for selecting RAID protection for IBM Spectrum Scale

Description	RAID 5	RAID 6	RAID 10	Advantage
Random write performance	2	3	1	RAID 10

Description	RAID 5	RAID 6	RAID 10	Advantage
Sequential write performance	1	2	3	RAID 5
Availability	3	1	2	RAID 6
Space efficiency	1	2	3	RAID 5
RAID level for IBM Spectrum Scale	Yes	Yes	No[a]	RAID5 and RAID6
Configuration for GFPS	4+P	8+2P	N/A	N/A
Minimum number of disks per LUNs	5	10	N/A	N/A

a. IBM Spectrum Scale uses efficient mechanisms for striping and mirroring the data. Although you can use storage R10, this would be more expensive from the capacity consumption perspective and less efficient for IBM Spectrum Scale, which natively distributes the data over multiple LUNs, while providing advanced data rebalancing capabilities. In specific cases, like Spectrum Scale metadata performance requirements, you can consider using LUNs on storage RAID 1 combined with striping performed by IBM Spectrum Scale.

Cache policy

This cache policy is an area of tuning where there are significant differences based on the style of storage server and the I/O workload. Generally, consider the following guidelines and apply it when supported by the storage subsystem:

- Sequential I/O workloads: Enable read and write cache, disable any read prefetch.
- Random I/O workloads: Enable read and write cache.
- Metadata disks: Enable read and write cache.

It may be more apparent why you would enable cache for random workloads but not so clear why you disable read cache prefetch on sequential workloads. When using IBM Spectrum Scale, we recommend disabling read prefetch mechanisms for the caching of data in the storage server. Although it works well for high concurrency environments and it can usually provide optimized performance, read-prefetch at the storage system should be disabled when using IBM Spectrum Scale because of its own mechanism for allocating the space, which also uses data prefetching. To a storage controller, the IBM Spectrum Scale read access pattern does not look sequential, so the read-ahead that is provided in most storage servers can degrade the read-performance. In fact, up to a 20% penalty can occur on certain models of storage.

Note: For availability reasons, most storage subsystems offer a write cache mirroring mechanism, between the storage controllers. Although disabling it might improve the write performance, if there is a power failure or storage controller failure, a risk of file system corruption exists.

Block size and calculation guidelines

To optimize disk I/O performance, consider the following options for NSD servers or other Spectrum Scale nodes that are directly attached to a SAN over a Fibre Channel network. When the storage server disks are configured for RAID 5, certain configuration settings can affect the Spectrum Scale performance. These settings are as follows:

- Spectrum Scale file system block size.
- Maximum I/O size of host FC HBA device driver.
- Storage server RAID 5 stripe size.

Figure 2-11 illustrates how to correlate the file system block, storage RAID level, and storage segment size when planning for the array configuration with IBM Spectrum Scale.

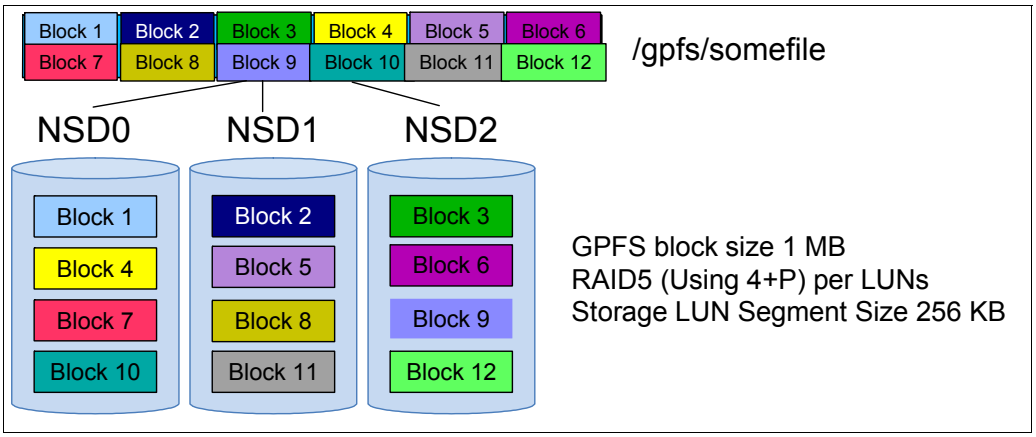


Figure 2-11 File system block size and storage LUN segment size guideline

For optimal performance, the file system block size should be a multiple of the maximum I/O size of the FC HBA device driver. In addition, the maximum I/O size of the FC HBA device driver must be a multiple of the RAID 5 stripe size. These suggestions might avoid the performance penalty of read-modify-write at the storage server for Spectrum Scale write operations. Examples of the suggested settings are as follows:

- 8+P RAID 5
  - File System block size = 512 KB.
  - Storage Server RAID 5 segment size = 64 KB (RAID 5 stripe size=512 KB).
  - Maximum I/O size of FC HBA device driver = 512 KB.
- 4+P RAID 5
  - File System block size = 256 KB.
  - Storage Server RAID 5 segment size = 64 KB (RAID 5 stripe size = 256 KB).
  - Maximum I/O size of FC HBA device driver = 256 KB.

For the example settings using 8+P and 4+P RAID 5, the RAID 5 parity can be calculated from the data written and prevents reading from disk to calculate the RAID 5 parity. The maximum I/O size of the FC HBA device driver can be verified by using the **`iostat`** command or the storage server performance monitor. In certain cases, the device driver might have to be patched to increase the default maximum I/O size.

I/O balance

I/O balance is key when you design a high performance Spectrum Scale cluster. This balance is most important when you are expecting to get the best sequential I/O performance from a solution. Therefore, you need to consider several aspects regarding how the Spectrum Scale I/O workload is distributed to the storage system:

- IBM Spectrum Scale reads and writes to all disks at the same time when performing sequential I/O operations. If the streaming operations are long enough, all the queues (adapter, disk, and so on) will fill up and start emptying at the same rate. This process slows the throughput of all the disks to that of the slowest disk.
- Mixing different types of disks or array types (for example, Raid5 and Raid6) in a file system, which participates in distributing the same I/O operations, should be avoided. You can use storage pools with mixed storage. Spectrum Scale storage pools allow you to keep disks of different performance levels inside a single file system without impacting the performance of the fastest disks. Such configurations are used with information lifecycle

management mechanisms to create storage tiers inside a Spectrum Scale file system where data can be migrated between tiers based on several criteria such as age of files or space thresholds.

- The same principles of balance apply to all levels of the I/O path including the Fibre Channel HBAs, host ports on the storage, and the number of arrays used for each controller.

Considerations for XIV storage with IBM Spectrum Scale

One of the benefits of the IBM XIV® Storage System is that there is no tuning required for most workloads, and this also applies to the Spectrum Scale storage configuration. When configuring a system with XIV and IBM Spectrum Scale, be sure multiple LUNS are available so you can have multiple SCSI devices available in the operating system for the best performance. For example, create 8 LUNs or 16 LUNS for a file system. In the XIV architecture, LUNs are spread over all available disks so there are no RAID configuration, segment size, or LUN layout considerations. Consider the following information:

- Be sure all server HBAs are zoned to see all XIV controller ports.
- Create multiple LUNS so that the operating system (Linux for example) can see multiple targets for better SCSI performance.
- Because XIV uses a 1 MB allocation size, for sequential data access, a Spectrum Scale file system block size of 1 MB can provide optimal performance.
- Use multiple LUNs in each Spectrum Scale file system so you can best use multiple SAN paths and device queues in the operating systems.

Solid-state drive (SSD) storage

The world of storage is no longer only about capacity; it is now also about the need for extremely fast and reliable access to data. The ever-increasing demand on today's servers, combined with the exponential growth of processor performance, has exposed the limitations of today's aging storage architectures.

IBM offers solid-state storage today with the new solid-state PCI Express (PCIe) adapters. Designed around silicon-based storage architecture, the PCI Express adapter has clustering technology, with performance comparable to DRAM and storage capacity on par with today's hard disks, giving you the power to improve both memory capacity and storage performance. Examples are as follows:

- SSD SAS, SATA interface
- Solid-state PCIe adapter type

The SSD-based storage system can be useful for applications exploiting the IBM Spectrum Scale solution. For example, IBM Spectrum Scale provides a storage pool architecture where it populates each file on special storage groups such as SSDs, HDDs, and SATA HDDs. Depending on how fast the store or retrieval speed is required, the fastest is the medium to where the file can be populated. Examples are as follows:

- Spectrum Scale metadata: If you use SSD for the metadata area of the file system, you can gain response time on the file system, especially concurrently accessing a group of files. A use case for this approach is for file servers, containing many files accessed concurrently by multiple users.
- SSD storage pool: This example is useful for RDBMS index database and real-time access database file.
- Spectrum Scale local cache: IBM Spectrum Scale 4.1 introduces the Local Read Only Cache (LROC) feature, which takes advantage of the SSD technology in order to improve the performance and reduce the load on the shared network and backend storage

devices. Many NAS workloads can benefit on the LROC facility including virtualization environments such as VMware and OpenStack, and also database environments. LROC is a new attribute used during the NSD creation. See 2.6.2, “Network Shared Disk creation considerations” on page 78.

4k sector support

The current disk technologies have been improved and most of the solid-state disks are now using a 4 k sector size. In order to cope with them, IBM Spectrum Scale adds support for using 4 k block disks devices in IBM Spectrum Scale 4.1.0.4. This capability is intended for all disks supporting 4 k sector size, however the main target is for SSD drives. For Spectrum Scale configurations where physical disks are managed by a RAID controller, this does not affect IBM Spectrum Scale, as long as the RAID controller continues to provide a traditional 512 byte sector interface in an efficient way.

The following implications apply to IBM Spectrum Scale due to 4 k alignment:

- Starting from 4.1, when a new file system is created, the default inode size is 4 k and the default block size is 256 k. The file system is by default 4 k aligned. All file systems created before IBM Spectrum Scale 4.1 are not 4 k aligned. These file systems are not migrated to 4 k alignment when the file systems are migrated to IBM Spectrum Scale 4.1. You need to re-create the file system using IBM Spectrum Scale 4.1 code. A possibility to migrate data for rebuilding the file system is using AFM background migration. See more details in Chapter 6, “Active File Management” on page 291.
- When adding disks to a file system with a smaller inode size and a block size explicitly specified, a warning message displays to show that the file system is not 4 k aligned. A native 4 k sector disk will not be able to be added to the file system later.

When adding disks to a 4 k aligned file system, there will not be any restrictions. For a non 4 k aligned file system, a warning message will be displayed for the 4 k disk, which supports 512 byte emulation. However, 4 k native disks can never be added to non 4 k aligned file systems.

2.3.4 Tape library

Although automated tape libraries have one or more tape drives, they are typically used with at least two tape drives. All tape cartridges are accessible to all drives, therefore making concurrent reading and writing operations possible. This section explains two types of tape drives for information lifecycle management (ILM). Usually, a single-port tape drive is used for simple backup; a dual-port tape drive is used for the ILM environment.

ILM with IBM Spectrum Scale is implemented using storage pools, which provide the capability to implement different storage tiers within a Spectrum Scale file system. By using external pools managed by Tivoli Storage Manager hierarchical storage management (HSM), the capability is extended to the use of Tivoli Storage Manager storage pools and tape devices.

Note: IBM Spectrum Scale is now able to attach the tape tier to a Spectrum Scale file system using the IBM Linear Tape File System™ Enterprise Edition (LTFS EE) support. For more details about deploying IBM Spectrum Scale with LTFS, refer to the following site:

`http://www.redbooks.ibm.com/abstracts/sg248143.html`

You can increase throughput by adding more drives. Libraries can exchange tapes in several seconds, substantially improving file-restore response times. Tape libraries also offer the feature of enabling a drive to take over if another drive fails.

Multi-drive automated tape libraries and ultra-scalable tape libraries combined with storage-management software, including concurrent backup, archive, and hierarchical storage management (HSM), offer the most robust solution to manage and protect huge amounts of corporate data. Automated tape libraries allow random access to large numbers of tape cartridges and the concurrent use of two or more drives, rather than manually loading one tape after another or using a single-drive sequential autoloader.

IBM offers the following tape library models: TS3100, TS3200, TS3310, TS3400, and TS3500. These models support two types of tape drives:

- Linear Tape-Open (LTO)
- TS1100 Series

Tape drives (LTO series)

Two LTO formats (Ultrium and Accelis) were originally introduced in 2000, and licenses for the new technology were made available. Since then, manufacturers have not pursued the Accelis format because the Ultrium format meets market needs. The LTO-sponsoring companies have taken steps to protect client investment by providing a six-generation roadmap to illustrate native capacity, shown in Figure 2-12, and establishing an infrastructure to enable compatibility between products.

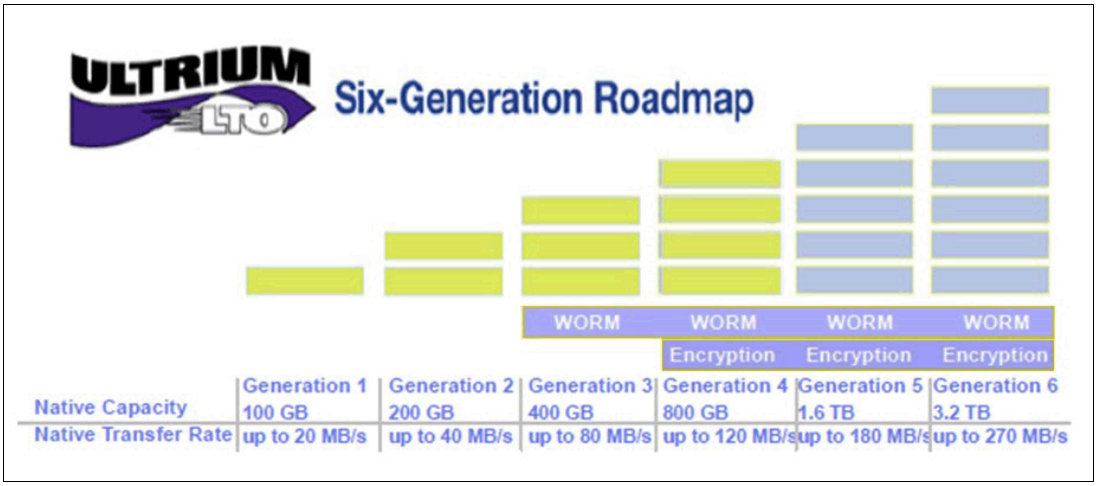


Figure 2-12 LTO Ultrium native capacity roadmap

TS1100 series tape drive

The IBM System Storage TS1120 to TS1150 tape drives offer a design of high capacity, performance, and high reliability for storing mission-critical data. With the 2003 introduction of the first generation of the new family of tape drives, IBM advanced its high-end half-inch cartridge tape technology. The latest generation of TS1100 drives series, TS1150 provides capability to write media with capacity up to 10 TB of decompressed data.

Data path failover provides a failover mechanism in the IBM device driver, enabling you to configure multiple redundant paths in a SAN environment. If there is a path or component failure, the failover mechanism automatically provides error recovery to retry the current operation using an alternate, pre-configured path without canceling the current job in progress. This capability allows you flexibility in SAN configuration, availability, and management.

LAN-free backup (SAN backup)

A technology provides an alternative path for data movement between the Tivoli Storage Manager client and the server. Shared storage resources (disk, tape) are accessible to both

the client and the server through the SAN. Data movement is offloaded from the LAN and from the server processor and allows for greater scalability. IBM Spectrum Scale can also be used for LAN-free operations, in an environment using TSM pools on disk systems, as LAN-free targets for client backup data. IBM Spectrum Scale in such configurations offers increased scalability, availability, and performance over the traditional file system sharing on SAN using IBM Tivoli SANergy®.

Figure 2-13 shows that the storage agent handles the communication with the Tivoli Storage Manager server over the LAN but sends the data directly to SAN-attached tape devices, relieving the Tivoli Storage Manager server from the actual I/O transfer.

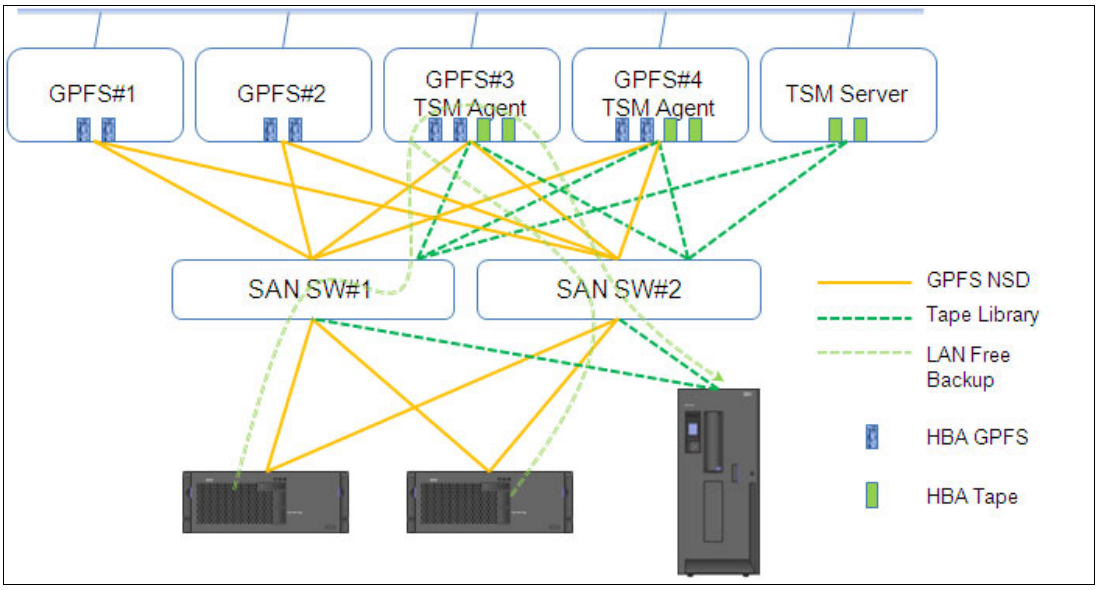


Figure 2-13 TSM LAN-free backup using IBM Spectrum Scale (formerly GPFS)

LAN-free backups decrease the load on the LAN by introducing a storage agent. The storage agent can be thought of as a small Tivoli Storage Manager server (without a database or recovery log) that is installed and runs on the Tivoli Storage Manager client machine.

2.4 IBM Spectrum Scale supported platforms

From the IBM Spectrum Scale point of view, all configuration commands in a mixed operating system environment are similar across platforms with some restrictions applied on Windows systems. In an AIX and Linux environment, the commands are almost the same. In a Windows system environment, not all the cluster commands are provided for the Spectrum Scale management functions. This section describes the supported operating systems and what Spectrum Scale functions are available in a mixed cluster.

IBM Spectrum Scale is supported on the following platforms:

- IBM AIX
- Linux on Power
- Linux on x86
- Linux on System z (new in IBM Spectrum Scale 4.1.0.5)
- Windows operating system

Table 2-3 summarizes the GPFS 3.5 and IBM Spectrum Scale 4.1 OS requirements.

Table 2-3 IBM Spectrum Scale operating system requirements by platform

Version	AIX	Linux on Power	Linux on x86_64	Linux on System z	Windows operating systems (x86 and x86_64)
IBM Spectrum Scale 4.1	AIX 7.1 AIX 6.1 TL4, or later	RHEL 6.2, or later[a] RHEL 7[b] SLES 11 SP2 or later[a] Ubuntu 14.04.1 (Power8 only)[c]	RHEL 6.2, or later SLES 11 SP2, or later Debian 7 Debian 6 Ubuntu 14.04.1	RHEL 6.5 RHEL 7.0 SLES 11 SP3	Windows Server 2008 x64 (SP2) Windows Server 2008 R2 Windows Server 2012 (Datacenter and Standard) Windows Server 2012 R2 (Datacenter and Standard) Windows 7 (Enterprise and Ultimate editions) x64 SP1 Windows 8.1 (Enterprise edition)
GPFS 3.5	AIX 7.1 AIX 6.1	RHEL 6 RHEL 5 SLES 11 SLES 10	RHEL 6 RHEL 5 SLES 11 SLES 10 Debian 6	N/A	Windows Server 2008 x64 (SP 2) Windows Server 2008 R2 Windows 7 (Enterprise and Ultimate editions) x64 SP1 Windows Server 2012 (Datacenter and Standard)[d]

[a] Big Endian.
[b] Big Endian, requires a minimum level of IBM Spectrum Scale 4.1.0.4.
[c] Little Endian, requires a minimum level of IBM Spectrum Scale 4.1.0.5.
[d] GPFS 3.5 supports Windows Server 2012 in homogeneous clusters only, with a minimum version of V3.5.0-11. IBM Spectrum Scale 4.1 does not have this restriction.

2.4.1 IBM Spectrum Scale on AIX

IBM Spectrum Scale runs with many different AIX versions and Technology Levels (TLs). We recommend applying the latest updates for the operating systems and drivers when using IBM Spectrum Scale.

Requirements and considerations for IBM Spectrum Scale on AIX:

- IBM Spectrum Scale FPO on AIX requires a minimum level of 3.5.0.11.
- Network File System (NFS) v4 in GPFS 3.5/IBM Spectrum Scale 4.1 is supported by AIX v6.1 and 7.1 versions.
- OpenSSL package included with certain releases of AIX v7.1 and 6.1 does not work with IBM Spectrum Scale because of a change in how the library is built. An appropriate version of OpenSSL must be installed. This version must be compiled with support for Secure Hash Algorithm (SHA).
- SCSI3 Persistent reservation can be used with IBM Spectrum Scale on AIX with storage systems that support persistent reservation, by either using SDDPCM software driver, or without SDDPCM with minimum AIX 6.1 TL7 SP4. Also, consult the storage documentation for additional considerations on drivers when enabling persistent reservation.

2.4.2 IBM Spectrum Scale for Linux (x86 and POWER)

Note: IBM Spectrum Scale 4.1 and GPFS 3.5 versions support only 64-bit Linux kernel.

Requirements and considerations for IBM Spectrum Scale on Linux are as follows:

- Latest versions of Linux and kernels (for example, RHEL 6.5, SLES 11SP3) require an updated version of IBM Spectrum Scale code level for versions before 4.1. Refer to the IBM Spectrum Scale FAQs web page for further details about the minimum IBM Spectrum Scale level required for a particular version of Linux and latest versions of kernels tested with IBM Spectrum Scale:

  http://ibm.co/1IKO6PN

- Kernel development files and compiler utilities are required to build the Spectrum Scale portability layer on Linux nodes. Following are the required packages for each supported Linux distribution:
  - SLES Linux RPMs: kernel-default-devel, cpp, gcc, gcc-c++, binutils.
  - RedHat Linux RPMs: kernel-devel, cpp, gcc, gcc-c++, binutils.
  - Debian Linux Packages: linux-headers, cpp, gcc, gcc-c++, binutils.
  - Ubuntu Linux Packages: linux-headers, cpp, gcc, gcc-c++, binutils.
- IBM Spectrum Scale 4.1 writes a GUID Partition Table (GPT) label on newly created NSDs, which makes it safer to use standard disk partitioning tools with Spectrum Scale NSDs. The previous versions of IBM Spectrum Scale 4.1 do not label a disk in a way that standard partitioning tools like **parted** or **fdisk** would recognize. In this case, the Linux administrator should verify the disk usage with IBM Spectrum Scale by using the **mmlsnsd** command to avoid modifying a disk that is part of the Spectrum Scale file systems.
- OpenSSL is required for remote cluster access.
- IBM Spectrum Scale for Linux on POWER does not support mounting a file system with a 16 KB block size when running on either RHEL 5 or SLES 11.
- Transparent Huge Page (THP) feature on RHEL 6.x is supported by IBM Spectrum Scale with the following minimum levels of kernels: RHEL 6.4 (kernel version 2.6.32-358.el6), RHEL 6.3 (kernel version 2.6.32-358.2.1), and RHEL 6.2 (kernel version 2.6.32-220.4.1.e;6.x86_64). On unsupported kernels, THP needs to be disabled at boot time, by using the following option: **transparent_hugepage=never**.
- IBM Spectrum Scale is not supported by the following Linux kernel types: RHEL hugemem, RHEL largesmp, RHEL uniprocessor (UP), SLES xen.
- IBM Spectrum Scale supports NFSV4 with the following distributions:
  - RHEL 5.5, and later.
  - SLES 11 SP1, and later service packs.

  Additional considerations when using NFSV4:
  - Package *nfs4-acl-tools* must be installed to support NFSv4 ACLs. The package can be found on the installation media of the Linux operating system.
  - Windows server-based NFSv4 clients are not supported by Linux/NFSv4 servers.
  - If a file system is to be exported over NFSv4/Linux, it must be configured to support POSIX ACLs (with -k all or -k posix options on **mmcrfs** and **mmchfs** commands).
  - Concurrent AIX/NFSv4 servers, Samba servers, and Spectrum Scale Windows operating system nodes in the cluster are allowed. NFSv4 ACLs may be stored in Spectrum Scale file systems via Samba exports, NFSv4/AIX servers, Spectrum Scale Windows operating system nodes, ACL commands of Linux NFSv3 and ACL commands of IBM Spectrum Scale.
- For cNFS, the following packages are required: *ethtool, nfs-utils, rpcbind*.

- Persistent Reservation (SCSI-3 PR) on Linux requires a minimum level of IBM Spectrum Scale 4.1, GPFS 3.5.0.1, or GPFS 3.4.0.10. If the storage system is capable of supporting the Activate Persist Through Power Loss (APTPL) feature on a Linux platform, you require GPFS 3.5.0.15, or later.
- When using IBM Spectrum Scale FPO on Linux on Power or x86, GPFS with minimum level of 3.5.0.7 is required, or later IBM Spectrum Scale 4.1 levels.
- Local read-only cache (LROC) feature is supported only on Linux x86_64 with IBM Spectrum Scale 4.1.0.1, or later updates.
- For the active file management (AFM) feature, the following package is required: *nfs-utils*.
- To use the **numaMemoryInterleave** parameter on x86 systems, the following package is required: *numactl*.

2.4.3 IBM Spectrum Scale for Linux on System z

IBM Spectrum Scale 4.1.0.5 introduces support for running Spectrum Scale file system on Linux on System z platforms. You can perform the administration in a manner similar to IBM Spectrum Scale on UNIX platforms (for example, you can add or delete disks while the file system is mounted).

Spectrum Scale administration commands are similar in name and function to UNIX and Linux operating system commands, with one important difference: the Spectrum Scale commands operate on multiple nodes.

The following requirements apply for IBM Spectrum Scale for Linux on System z:

- Hardware: IBM Spectrum Scale for Linux on System z, V4.1 is supported on IBM System z servers. Following are the minimum hardware requirements:
  - CP or IFL processors
  - 1 GB of system memory

  IBM Spectrum Scale for Linux on System z supports IBM ECKD™ DASDs and Fibre Channel Protocol (FCP) attached Small Computer System Interface (SCSI) disks. Additionally, see the FAQs for supported connectivity and qualified disk subsystems:

  http://ibm.co/1fRXoxg
- Software: You are required to use IBM Spectrum Scale 4.1.0.5, or later version for Linux on System z. The minimum operating system requirements are shown in Table 2-4.

Table 2-4 Supported Linux versions for IBM Spectrum Scale on System z

Linux Version	Fix level	Kernel level
SUSE Linux Enterprise Server 11	SP3 +Maintweb Update or later	3.0.101-0.15-default
Red Hat Enterprise Linux 7		3.10.0-123.6.3.el7.s390x
Red Hat Enterprise Linux 6.5	Update RHSA-2014-0328 or later	2.6.32-431.11.2.el6

Service updates for Linux: The latest service updates are mandatory for RHEL 6.5 and SLES 11 SP3. For all environments, we recommend that you install the latest service update. You can download it from IBM Fix Central:

http://www-933.ibm.com/support/fixcentral

IBM Spectrum Scale has no dependency on a specific version of z/VM.

IBM Spectrum Scale for Linux on System z uses a portability layer (kernel modules) that enables the Spectrum Scale daemon to interact with the Linux kernel. During the installation, you have to build the portability layer on your Linux instance, which fits in a wide variety of Linux Kernel versions and configurations.

IBM Spectrum Scale for Linux on System z can work in two modes:

- Shared disk (SAN) mode
- Network Shared Disk (NSD) mode

Shared disk mode

In this type of configuration, all of the nodes in the Spectrum Scale cluster are connected to a common set of disks. The disk I/O can perform better because all the nodes connect to the storage servers directly. This configuration mode is available only in homogeneous platforms. It is not supported on heterogeneous configurations.

Network Shared Disk mode

You can configure a Spectrum Scale cluster in which some nodes attach directly to the disks and other nodes access the Spectrum Scale server node disks through the network. This configuration is usually used in large clusters or to provide a cost-effective, potential high-performance solution.

When the Spectrum Scale node provides access to a disk for another node in the cluster, it is called an *NSD server*. The Spectrum Scale node that is accessing the data through an NSD server is called an *NSD client*. In this scenario, the NSD servers are connected to storage servers directly, and the NSD client accesses the file system through a high-speed network connecting to NSD servers.

At this point, IBM Spectrum Scale for Linux on System z is only supported by the storage subsystems that are shown in Table 2-5.

Table 2-5 Supported storage subsystems for IBM Spectrum Scale on Linux for System z

Storage system	SCSI device	ECKD device
IBM DS8000 series	Disk leasing or persistent reserve	Disk leasing
IBM Storwize V7000	Disk leasing or persistent reserve	N/A
IBM XIV	Disk leasing or persistent reserve	N/A
IBM FlashSystem™	Disk leasing or persistent reserve	N/A

Additional considerations for IBM Spectrum Scale for Linux on System z

Consider the following aspects when using IBM Spectrum Scale for Linux on System z:

- Currently, functions are limited to the Express Edition of IBM Spectrum Scale. See also the IBM Spectrum Scale FAQs web page.
- IBM Spectrum Scale for Linux on System z, V4.1 is supported only in IBM WebSphere® Application Server, WebSphere MQ, or similar workload infrastructure environments. Consult IBM Spectrum Scale FAQs for additional updates.
- Heterogeneous clusters are supported without local storage access but are limited to client-only POWER servers running the AIX or Linux operating system, or x86 clients running Linux.
- Only 64-bit operating system applies.
- IBM Spectrum Scale for Linux on System z, V4.1 is supported only with IBM System Storage DS8000, IBM Storwize V7000, IBM FlashSystem, or IBM XIV.
- The maximum number of nodes in a cluster with IBM Spectrum Scale on Linux/System z is 32. Consult IBM Spectrum Scale FAQs for any additional updates.

2.4.4 IBM Spectrum Scale on Windows operating systems

IBM Spectrum Scale can be used with Windows operating systems as both client or server nodes in a cluster of Windows operating system-only nodes, or in a mixed cluster with Linux or AIX nodes.

Prerequisites for IBM Spectrum Scale on Windows

- IBM Spectrum Scale V4.1 no longer requires the Subsystem for UNIX-based Applications (SUA) feature in Windows. This feature has been completely removed beginning with Windows Server 2012 R2 (Windows 8.1). IBM Spectrum Scale V4.1 removes SUA as a prerequisite, instead IBM Spectrum Scale now requires Cygwin from Red Hat (`http://www.cygwin.com`). Cygwin 32-bit (version 1.7 or later) must be installed before installing IBM Spectrum Scale V4.1 on Windows. Cygwin 64-bit is currently not supported (consult IBM Spectrum Scale FAQs for any additional updates). Ensure that the following packages will be installed at a minimum:
  - flip: Convert text file line endings between UNIX and DOS formats
  - m4: GNU implementation of the traditional UNIX macro processor
  - mksh: MirBSD Korn Shell
  - perl: Larry Wall's Practical Extracting and Report Language
  - procps: System and process monitoring utilities
  - openssh: The OpenSSH server and client programs (optional, required if you plan on mixing Linux, AIX, and Windows nodes in the same cluster)

  Rolling upgrade of GPFS V3.5 (SUA based) to IBM Spectrum Scale V4.1 (Cygwin based) is supported. Adding a new IBM Spectrum Scale V4.1 Windows node to an existing cluster having V3.5 nodes is also supported.
- IBM Spectrum Scale requires the use of OpenSSH to support its administrative functions only when the cluster includes Windows nodes and UNIX nodes. Clusters that only include Windows nodes do not require OpenSSH. They can use the **`mmwinrsh`** utility that comes with IBM Spectrum Scale. The fully-qualified path name is **`/usr/lpp/mmfs/bin/mmwinrsh`**.

Follow these steps to install OpenSSH:

- In IBM Spectrum Scale 4.1, OpenSSH package is installed with Cygwin.
- For GPFS 3.5 you can download it from IBM Fix Central:

  http://ibm.co/1IRCOsO

- User Account Control (UAC) must be disabled for Windows Server 2008 (32 bit). Although it is not a mandatory requirement for the rest of supported versions, we recommend disabling UAC to avoid any possible impact on some Spectrum Scale functions.
- If Windows Firewall is enabled, you need to modify the default configuration. See the ports used for Spectrum Scale daemon communication in the section “Security considerations for IBM Spectrum Scale clusters” on page 69.
- Spectrum Scale diagnostic tracing (**mmtracectl**) on Windows operating systems uses the Microsoft programs called **tracefmt.exe** and **tracelog.exe**. They are not mandatory for the Spectrum Scale functionality, but we recommend installing them for problem determination purposes. These programs are not included with Windows operating systems, but they are part of the Windows Driver Kit (WDK), which can be downloaded from Microsoft:

  http://msdn.microsoft.com/en-us/windows/hardware/hh852365

- Spectrum Scale nodes must be part of an Active Directory domain. IBM Spectrum Scale expects that all Windows operating system nodes in a cluster are members of the same domain. Join the nodes in the domain before configuring the Spectrum Scale cluster.

IBM Spectrum Scale considerations on Windows operating systems

IBM Spectrum Scale for Windows operating systems does not fully support all of the Spectrum Scale features that are available on AIX and Linux. Some of these limitations constrain how you can configure a Spectrum Scale cluster when it includes Windows operating system nodes. The remaining limitations only pertain to Windows operating system nodes rather than the whole cluster.

The following restrictions apply on Windows servers:

- Commands not supported:

  mmafmctl, mmafmconfig, mmafmlocal, mmapplypolicy, mmbackup, mmbackupconfig, mmrestoreconfig, mmcheckquota, mmdefedquota, mmedquota, mmlsquota, mmrepquota, mmclone, mmdelacl, mmeditacl, mmgetacl, mmputacl, mmimgbackup, mmimgrestore, mmpmon.

  In IBM Spectrum Scale V4.1, the following user commands require administrative privileges. They can only be run by a user who is a member of the Administrators group: **mmchfileset**, **mmcrsnapshot**, **mmdelsnapshot**, **mmdf**, **mmlsdisk**, **mmlsfileset**, **mmlsfs**, **mmlspolicy**, **mmlspool**, **mmlssnapshot**, and **mmsnapdir**. This restriction does not apply to GPFS v3.5. Ordinary users can execute the preceding commands in the SUA environment.
- Exporting the Spectrum Scale file systems as Server Message Block (SMB) shares (also known as CIFS shares), or NFS serving (any version of NFS) by Spectrum Scale Windows operating system nodes is not supported.
- The Spectrum Scale application programming interfaces (APIs) are not supported on Windows operating system.
- The native Windows operating system backup utility is not supported.
- IBM Spectrum Scale for Windows operating system is not supported in any environment where Citrix Provisioning Services are deployed.

- Symbolic links that are created on UNIX based nodes are specially handled by Spectrum Scale Windows operating system nodes. They appear as regular files with a size of zero and their contents cannot be accessed or modified.
- IPv4 subnets are not supported on a cluster that is defined with IPv6 primary addresses (hostname) that contain Windows operating system nodes.
- In a multicluster environment, a Spectrum Scale cluster can mount a file system owned by another cluster. OpenSSL must be installed on all nodes. In such environments using Windows servers, only the AUTHONLY cipher can be used for authentication. Refer to the **mmauth** command for further details about setting or changing the authentication in a Spectrum Scale multicluster environment.
- With IBM Spectrum Scale V4.1, GPFS V3.5, and GPFS V3.4.0-14 (and later), a DMAPI-enabled file system can be mounted on a Windows operating system node, with certain restrictions. See the following IBM Knowledge Center site:

  IBM Tivoli Storage Manager Version 6.3:

  http://ibm.co/1yHxQsP

  Tivoli Storage Manager support page:

  http://www-01.ibm.com/support/docview.wss?rs=663&tc=SSGSG7&uid=swg21248771

Virus scan and monitor program on Windows Spectrum Scale clusters

If more than one Spectrum Scale Windows operating systems node is running antivirus software that scans directories and files, shared files only need to be scanned by one Spectrum Scale node. Scanning shared files more than once is not necessary. When you run antivirus scans from more than one node, schedule the scans to run at separate times to allow better performance of each scan, and to avoid any conflicts that might arise because of concurrent exclusive access attempts by the antivirus software from multiple nodes.

Notes:

- Enabling real-time antivirus protection for Spectrum Scale volumes might significantly degrade the performance and cause excessive resource consumption.
- Consider using a single, designated Windows operating systems node to perform all virus scans.

Difference between IBM Spectrum Scale and NTFS

IBM Spectrum Scale differs from the Microsoft Windows NT File System (NTFS) in its degree of integration into the Windows system administrative environment, Windows Explorer, and the desktop. The differences are as follows:

- Manual refreshes are required to see any updates to the Spectrum Scale namespace.
- You cannot use the recycle bin.
- You cannot use distributed link tracking. This is a technique through which shell shortcuts and OLE links continue to work after the target file is renamed or moved. Distributed link tracking can help you locate the link sources in case the link source is renamed or moved to another folder on the same or different volume on the same computer, or moved to a folder on any computer in the same domain.
- You cannot use NTFS change journaling. This also means that you cannot use the Microsoft Indexing Service or Windows Search Service to index and search files and folders on Spectrum Scale file systems.

Additionally, consider that the following NTFS features are not supported by Spectrum Scale file systems on Windows operating systems:

- File compression (on individual files or on all files within a folder).
- Encrypted directories or encrypted files.
- Windows NTFS quota management (Spectrum Scale quotas are administered through Spectrum Scale specific commands).
- Reparse points.
- Defragmentation and error-checking tools.
- Alternate data streams (ADSs).
- The assignment of an access control list (ACL) for the entire drive.
- A change journal for file activity.
- The scanning of all files or directories that a particular SID owns (FSCTL_FIND_FILES_BY_SID).
- Generation of AUDIT and ALARM events specified in a System Access Control List (SACL). IBM Spectrum Scale is capable of storing SACL content, but does not interpret it.
- Windows operating system sparse files API.
- Transactional NTFS (also known as TxF).

Identity Management on Windows operating systems

IBM Spectrum Scale allows file sharing among AIX, Linux, and Windows operating system nodes. AIX and Linux rely on 32-bit user and group IDs for file ownership and access control purposes. Windows systems use variable-length security identifiers (SIDs). The difference in the user identity description models presents a challenge to any subsystem that allows for heterogeneous file sharing.

IBM Spectrum Scale uses 32-bit ID name space as the canonical name space, and Windows SIDs are mapped into this name space as needed. Two mapping algorithms are used (depending on system configuration):

- IBM Spectrum Scale built-in auto-generated mapping.
- User-defined mappings that are stored in the Microsoft Windows Active Directory using the Microsoft Identity Management for UNIX (IMU) component.

2.4.5 Summary of IBM Spectrum Scale support functions on operating systems

Table 2-6 lists the functions that are supported on each operating system (OS). The following functions and the support statements are referenced to GPFS 3.5 and IBM Spectrum Scale 4.1 versions.

Table 2-6 IBM Spectrum Scale support functions on each operating system

Function	AIX	Linux (x86 and Power)	Linux on System z (*)	Windows operating system	Multiplatform cluster (**)
Persistent Reservation	Yes	Yes	Yes[g]	No	Yes[c]
DMAPI	Yes	Yes	No	Yes[a]	Yes[a]

Function	AIX	Linux (x86 and Power)	Linux on System z (*)	Windows operating system	Multiplatform cluster (**)
ILM/HSM	Yes	Yes[e]	No	No	Yes[b,e]
ILM/LTFS EE	No	Yes[e]	No	No	Yes[b,e]
InfiniBand RDMA	No	Yes	No	No	Yes[b]
InfiniBand IPoIB	Yes	Yes	No	Yes	Yes
NSD: LUN (block device)	Yes	Yes	Yes	Yes	No[c]
NSD: server/client	Yes	Yes	Yes	Yes	Yes
Multipath driver	MPIO	Device Mapper	Device Mapper(FCP)	MPIO	No[c]
mmbackup	Yes	Yes	No	No	Yes[b]
mmquota	Yes	Yes	Yes	No	Yes[b]
mmacl	Yes	Yes	Yes	No	Yes[b]
mmpmon	Yes	Yes	Yes	No	Yes[b]
Snapshot support	Yes	Yes	Yes	Yes	Yes
GPFS APIs (Programming Interfaces)	Yes	Yes	Yes	No	Yes[b]
Default administrator name	root	root	root	administrator[d]	root
Active File Management	Yes[f]	Yes	No	No	Yes[b]
Encryption (IBM Spectrum Scale 4.1 Advanced Edition, or later)	Yes	Yes	No	No	Yes[b]

[*] IBM Spectrum Scale support for Linux on System z requires version 4.1.0.5, or later. The first version is based on IBM Spectrum Scale 4.1 Express Edition, which includes most of the base level features. IBM intends to offer additional functionality that is in the Standard and Advanced Editions in future versions.

[**] The current version of IBM Spectrum Scale for Linux on System z can support up to 32 nodes and a heterogeneous cluster in Network Shared Disk (NSD) mode. In a heterogeneous cluster, the NSD server must be Linux on System z and the NSD clients (without direct storage access) can be: Linux on Power, Linux on x86, or AIX.

[a] Restrictions apply for Windows platforms using DMAPI (see “IBM Spectrum Scale considerations on Windows operating systems” on page 62).

[b] Function not supported on Windows operating system nodes within a multiplatform cluster. In case of InfiniBand RDMA function, only Linux (x86 and Power) nodes are supported by IBM Spectrum Scale in a multiplatform cluster. For Encryption case: it should be disabled in a cluster with Windows operating system nodes. For AFM case: application nodes can be either AIX or Linux; gateway nodes are only supported by the Linux operating system.

[c] NSD servers in a multiplatform cluster need to be the same platform. Concurrent LUN access from different platforms is not supported by IBM Spectrum Scale.

[d] On multiplatform cluster, use root user; IBM Spectrum Scale 4.1 requires Cygwin environment on Windows system nodes. Using Identity Management for UNIX with Active Directory is

optional.

[e] Not supported for Linux on Power Systems. For ILM with HSM, HSM is available only on AIX and Linux x86_64. LTFS EE is supported only on Linux x86_64.

[f] AFM support for AIX nodes requires GPFS with minimum level 3.5.0.11, or later IBM Spectrum Scale levels. AIX nodes can be only application nodes.

[g] Only supported by SCSI devices (not for ECKD).

2.4.6 GPT partition table

IBM Spectrum Scale 4.1 introduces a new NSDv2 format for the Spectrum Scale disks. When creating a new NSD (**mmcrnsd**) in a Spectrum Scale 4.1 cluster on Linux systems, a GPT partition table is written on the NSD to prevent the inadvertent use of the NSD by administrators and system utilities. It also improves the Spectrum Scale device discovery and allows standard tools to identify the Spectrum Scale disks.

The Spectrum Scale commands and utilities do recognize NSD devices and do not overwrite data unless the user uses the force flag for the operation. For instance, any attempt to add an NSD to a file system when it is already part of a Spectrum Scale file system will be blocked. However, running a standard Linux disk partitioning command like **parted** against an existing Spectrum Scale NSD results in the following:

```
# parted /dev/sdy mklabel gpt
Warning: The existing disk label on /dev/sdy will be destroyed and all data on
this disk will be lost.
Do you want to continue?
Yes/No? n
```

Note: Due to a potential exposure observed when using UEFI firmware, which in certain conditions might overwrite the Spectrum Scale GPT table with an existing backup GPT table, we recommend using IBM Spectrum Scale 4.1.0.2 code level, or later. At this level of code, at creation of NSDs, both primary and secondary copies of old GPT disks are overwritten by IBM Spectrum Scale. However, if NSDs were created in previous 4.1 levels, they need to be checked if they contain a secondary old GPT table.

An **nsdcheck** script is available to run against NSD devices to determine if there is a valid backup GPT table on the device. An NSD disk is at risk if the remarks display *hasPrimaryGpt=no, hasSecondaryGpt=yes*. If the backup table is not valid, the script can then be used to clear the backup GPT table on the NSD device, before any firmware updates. Running this script is recommended for all Linux deployments, as a precaution. When used to remove the secondary GPT, the script will only remove this if there is a GPT signature on the last sector of the NSD device but not at the beginning.

The script is shipped in the `gpfs.base` package, in the `/usr/lpp/mmfs/samples` directory of GPFS V3.4.0.29, V3.5.0.19 or later versions. Fixes can be downloaded from IBM Fix Central:

http://www-933.ibm.com/support/fixcentral

2.4.7 SCSI-3 Persistent Reservation

SCSI-3 Persistent Reservation (PR) can be used with IBM Spectrum Scale to provide faster disk failover times when the underlaying storage supports this feature. SCSI3 PR allows the stripe group manager (also known as *file system manager*) to “fence” disks during node failover by removing the reservation keys for that node. In contrast, non-PR disk failover causes the system to wait until the disk lease expires.

IBM Spectrum Scale allows file systems to have a mix of PR and non-PR disks. In a mixed configuration, IBM Spectrum Scale will fence PR disks for node failures and recovery and non-PR disk will use disk leasing. We recommend that all disks in a file system use PR (if supported).

Considerations for Persistent Reservation with IBM Spectrum Scale

When planning for use of IBM Spectrum Scale with disks supporting PR, consider the following:

- Only AIX or Linux servers are supported by SCSI3-PR. All NSD servers or direct attached nodes must be AIX or Linux. A mixed environment with Persistent Reservation is not supported. Requirements:
  - IBM Spectrum Scale support for Persistent Reservation on Linux requires a minimum level of IBM Spectrum Scale v4.1, GPFS v3.5.0-1, or GPFS v3.4.0-10.
  - GPFS v3.4 on IBM AIX 5L™ v5.3 requires APARS IZ01534, IZ04114, and IZ60972.
  - GPFS v3.4 or v3.5 support on AIX v6.1 requires APAR IZ57224.
  - AIX v6.1 TL7 + Service Pack 4 is required to support Persistent Reserve without SDDPCM.
  - For GPFS v3.4.0-10, v3.5, or IBM Spectrum Scale v4.1 support on AIX v7.1, refer to the storage documentation to install the correct multipath device driver.
- The use of Persistent Reservation is now supported on tie-breaker disks with a minimum level of GPFS v3.5.0.21, or later and IBM Spectrum Scale v4.1.0.4, or later.
- For the Activate Persist Through Power Loss (APTPL) feature:
  - On Linux, if the storage is capable of supporting APTPL, GPFS v3.5.0.15 or later supports this feature.
  - On AIX, your level of multipath driver must support APTPL.
- Starting with 3.5.0.16, it is possible to have a descOnly disk that resides on a device that does not support SCSI-3 Persistent Reservation while allowing Persistent Reservation to be used on other disks in the same file system. The lack of Persistent Reservation support for the descOnly disk does not result in fast failover being disabled.
- For a detailed list of storage systems and drivers supported for PR with IBM Spectrum Scale, consult IBM Spectrum Scale FAQs:

  http://ibm.co/1IKO6PN

Enabling IBM Spectrum Scale to use Persistent Reservation

Before enabling the SCSI3-PR in IBM Spectrum Scale, verify the disks planned to be used with IBM Spectrum Scale for the PR capability:

- In AIX, use the **`lsattr -R`** command to determine the supported reservation modes for an hdisk device and check that **PR_shared** mode is present. See Example 2-12.

Example 2-12 Determining the reservation modes supported by a disk device in AIX

```
root@fsaix1:/>lsattr -R -l hdisk1 -a reserve_policy
no_reserve
single_path
PR_exclusive
PR_shared
```

- In Linux, you can use the **`sg_persist`** utility (part of sg3_utils package) or **`mpathpersist`** only with DM multipath devices (part of device-mapper-multipath package) to query and

manage PR reservations on disks. Example 2-13 shows two outputs for a device /dev/sda, which is a non-PR disk and for a multipath device /dev/mapper/mpathb, which is PR capable.

Example 2-13 sg_persists output for two types of disks in Linux

```
[root@fslinux1 gpfs]# sg_persist -d /dev/sda
>> No service action given; assume Persistent Reserve In command
>> with Read Keys service action
  AIX       VDASD             0001
  Peripheral device type: disk
PR in: command not supported

[root@fslinux1 gpfs]# sg_persist -d /dev/mapper/mpathb
>> No service action given; assume Persistent Reserve In command
>> with Read Keys service action
  IBM       2145              0000
  Peripheral device type: disk
  PR generation=0x0, there are NO registered reservation keys
```

To enable PR use in IBM Spectrum Scale, use the following command:

```
mmchconfig usePersistentReserve=yes
```

Note: Changing the *usePersistentReserve* attribute requires that IBM Spectrum Scale is stopped on all cluster nodes.

If the disks are already configured as NSDs in the Spectrum Scale file systems or new NSDs are added, they are automatically configured for PR. If the **mmchconfig** command fails, the most likely cause is either a hardware or device driver problem. Other PR-related errors will probably be seen as file system unmounts that are related to disk reservation problems. This type of problem should be debugged with existing trace tools.

To verify the persistent reservation configuration, use the following commands:

- Verify the option usePersistentReserve is set to yes using **mmlsconfig**. See Example 2-14.

Example 2-14 mmlsconfig output

```
root@fsaix1:/>mmlsconfig
Configuration data for cluster tsmgpfs.fsaix1:
----------------------------------------------
clusterName tsmgpfs.fsaix1
clusterId 12742445374757012402
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
usePersistentReserve yes
ccrEnabled yes
autoload yes
tiebreakerDisks tie1
adminMode central
......
```

- Verify the disk is used in PR mode using the **mmlsnsd -X** command. See Example 2-15.

Example 2-15 Verifying persistent reservation use on NSD level

```
root@fsaix1:/gpfs>mmlsnsd -X

 Disk name    NSD volume ID      Device         Devtype  Node name                Remarks
---------------------------------------------------------------------------------------------------
 gpfs2nsd     AC1O14995436CE37   /dev/hdisk9    hdisk    fsaix1                   pr=yes
 gpfs5nsd     AC1O149B543849BC   /dev/hdisk2    hdisk    fsaix1                   pr=yes
 gpfs6nsd     AC1O149B543849BD   /dev/hdisk3    hdisk    fsaix1                   pr=yes
 gpfs7nsd     AC1O149B54384DO9   /dev/hdisk5    hdisk    fsaix1                   pr=yes
 gpfs8nsd     AC1O14995438500 7  /dev/hdisk4    hdisk    fsaix1                   pr=yes
 gpfs9nsd     AC1O149B543857AD   /dev/hdisk7    hdisk    fsaix1                   pr=yes
 tie1         AC1O14995438B30C   /dev/hdisk1    hdisk    fsaix1
```

mmlsnsd -X returns the local devices status for the node where the command is run. Check this command on all cluster nodes to verify that the activation of PR is complete for a disk. Also observe in Example 2-15 that the tiebreaker disk is not set to PR mode.

- To view the keys that are currently registered on a disk, issue the **tsprreadkeys** command from a node that has access to the disk. See Example 2-16.

Example 2-16 Query the PR keys from a disk device

```
root@fsaix1:/gpfs>/usr/lpp/mmfs/bin/tsprreadkeys hdisk2
Registration keys for hdisk2
1. 00006d0000000004
2. 00006d0000000004
3. 00006d0000000003
4. 00006d0000000003
5. 00006d0000000001
6. 00006d0000000001
7. 00006d0000000002
8. 00006d0000000002
9. 00006d0000000002
10. 00006d0000000002
11. 00006d0000000001
12. 00006d0000000001
13. 00006d0000000003
14. 00006d0000000003
15. 00006d0000000004
16. 00006d0000000004
```

Observe that there are 16 values returned corresponding with four distinct key values (the nodes) x 4 paths each node to the device hdisk2.

2.5 Security considerations for IBM Spectrum Scale clusters

This section describes the security mechanism that IBM Spectrum Scale uses to manage the cluster.

2.5.1 Remote shell with ssh

Spectrum Scale commands need to be able to communicate across all nodes in the cluster. To achieve this, the Spectrum Scale commands use the remote shell command that you specify on the **mmcrcluster** command or the **mmchcluster** command.

The default remote shell command is **rsh**. You can designate the use of a different remote shell command by specifying its fully qualified path name with the **mmcrcluster** command or the **mmchcluster** command. The remote shell command that you use must adhere to the same syntax as the **rsh** command, but can implement an alternate authentication mechanism. Clusters that include both UNIX and Windows operating system nodes require the use of **ssh** for the remote shell command.

By default, you can issue Spectrum Scale administration commands from any node in the cluster. Optionally, you can choose a subset of the nodes that are capable of running administrative commands. In either case, the nodes that you plan to use for administering the Spectrum Scale cluster must be able to run remote shell commands on any other node in the cluster as a “root” user without the use of a password and without producing any extraneous messages.

2.5.2 Remote cluster: Subnets, firewall rules, and TCP port numbers

IBM Spectrum Scale allows you to share data across multiple Spectrum Scale clusters. After a file system is mounted in another Spectrum Scale cluster, all access to the data is the same as though you were on the host cluster. You can connect multiple clusters within the same data center or across long distances over a WAN. In a multicluster configuration, each cluster can be placed in a separate administrative group simplifying administration or provide a common view of data across multiple organizations.

The firewall setting must allow the following rule (the network switch-based firewall rule setting should be the same as the operating system firewall):

- Inbound/outbound, GPFS/1191 TCP
- Inbound/outbound, SSH/22 TCP

Note: For Linux, you might need to configure iptables on the Spectrum Scale nodes to allow communication on these ports because by default, the firewall rules will not permit. Check if your iptables service is enabled in your environment and the current rules applied using: **service iptables status**.

IBM Spectrum Scale uses by default the port number 1191 for the daemon communication. You can change this port by using the **mmchconfig tscTcpPort=NewPortNumber** command. When you change the main port (daemon port) number, you must also change the mmsdrserv port to the same number by using the **mmchconfig mmsdrservPort=NewPortNumber** command.

Certain commands (**mmadddisk**, **mmchmgr**, and so on) require an additional socket to be created during the command. The port numbers assigned to these temporary sockets are controlled with the *tscCmdPortRange* configuration parameter. If an explicit range is not specified, the port number is dynamically assigned by the operating system from the range of ephemeral port numbers. If you want to restrict the range of ports used by Spectrum Scale commands, use the **mmchconfig** command:

```
mmchconfig tscCmdPortRange=LowNumber-HighNumber
```

To specify a port number when connecting to remote clusters, use the **mmremotecluster** command.

```
mmremotecluster update ClusterName -n tcpPort=PortNumber,Node,Node...
```

Figure 2-14 illustrates a Spectrum Scale remote-cluster mount diagram.

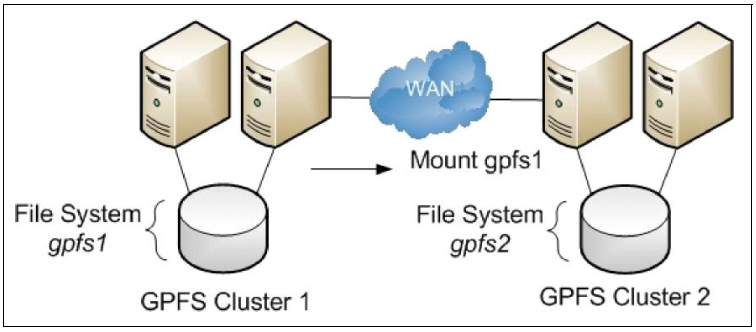


Figure 2-14 Multi-cluster configuration

2.5.3 SELinux configuration with IBM Spectrum Scale

You can run IBM Spectrum Scale in a Security Enhanced Linux (SELinux) environment. However, several considerations apply for Spectrum Scale services and file systems.

Spectrum Scale initscript

When running Spectrum Scale on SELinux, you cannot allow the normal init scripts to start Spectrum Scale because the script causes the Spectrum Scale daemons to run in the SELinux domain “initrc_t”, which is too restricted for Spectrum Scale. To get Spectrum Scale to run fully unconfined, use the **runcon** command to set the security context. Run the following command in `/etc/rc.d/rc.local`:

```
runcon -t unconfined_t /usr/lpp/mmfs/bin/mmstartup
```

Then, disable automatic start of Spectrum Scale by running the following command:

```
chkconfig gpfs off
```

File system labels

The use of SELinux file labels is not supported before GPFS 3.5. Running older versions of GPFS with SELinux enabled (in either permissive or enforcing mode) and inode labels stored as xattrs has a significant risk of triggering deadlocks. This scenario is particularly true under heavy workloads and when Spectrum Scale admin commands that quiesce file system activity are issued (for example, **mmcrsnapshot** or **mmdelsnapshot**).

Starting with GPFS 3.5, the use of SELinux inode labels is supported. However, there is a limitation: If a security label is changed after the object is created, there is no guarantee that the change will be immediately visible on other nodes (the change will be visible in most cases, but not all, if the inode in question is in use (for example: it belongs to an open file), the stale cached security state remains). This problem has to do with the lack of inode security state invalidation API in the Linux kernel, and affects other cluster file systems, not just Spectrum Scale.

One possible alternative to inode labeling is to set a label on the file system at mount time. Any files in the Spectrum Scale file system will then have the same label. For example, to mount a Spectrum Scale file system with xen-images, disable normal Spectrum Scale automount of the file system by running the following command:

```
mmchfs gpfsxen -A no
```

Then, “manually” mount the file system with the correct fscontext by adding the following command to `/var/mmfs/etc/mmfsup.scr`:

```
mount /dev/gpfsxen -t gpfs -o
"fscontext=system_u:object_r:xen_image_t",rw,mtime,atime,dev=gpfsxen
/var/lib/xen/images
```

Impact on file system backup operations

Enabling SELinux also impacts file system backup due to inode labels that are stored as xattrs. For example, once a file is enabled for SELinux and xattrs are added, Tivoli Storage Manager incremental backup picks this file for backup. Data and metadata are backed up again regardless of any other changes to the file.

For more details, see the following web page:

http://ibm.co/1EGrVeV

2.6 IBM Spectrum Scale configuration planning

This section provides information regarding IBM Spectrum Scale availability features, NSDs, and file system considerations in a Spectrum Scale cluster.

2.6.1 High availability

This section describes the high availability characteristics of a Spectrum Scale cluster.

Node quorum

A Spectrum Scale cluster is formed by a collection of nodes that share access to the file systems defined in the cluster. A node is an independent operating system instance residing on a physical machine, or on a virtual machine (for example, an AIX or Linux LPAR on a Power system or a VM in a VMware ESX environment). IBM Spectrum Scale uses a cluster mechanism called *quorum* to maintain data consistency if there is a node failure.

Quorum operates on the principle of majority rule. This means that a majority of the quorum nodes in the cluster must be successfully communicating before any node can mount and access a file system. The quorum keeps any nodes that are cut off from the cluster (for example, by a network failure) from writing data to the file system.

During node failure situations, quorum must be maintained so that the cluster can remain online. If quorum is not maintained because of node failure, IBM Spectrum Scale unmounts local file systems on the remaining nodes and attempts to reestablish quorum, at which point file system recovery occurs. For this reason, be sure to carefully consider the set of quorum nodes.

Figure 2-15 shows both configurations: quorum-only nodes and quorum nodes with tiebreaker disks.

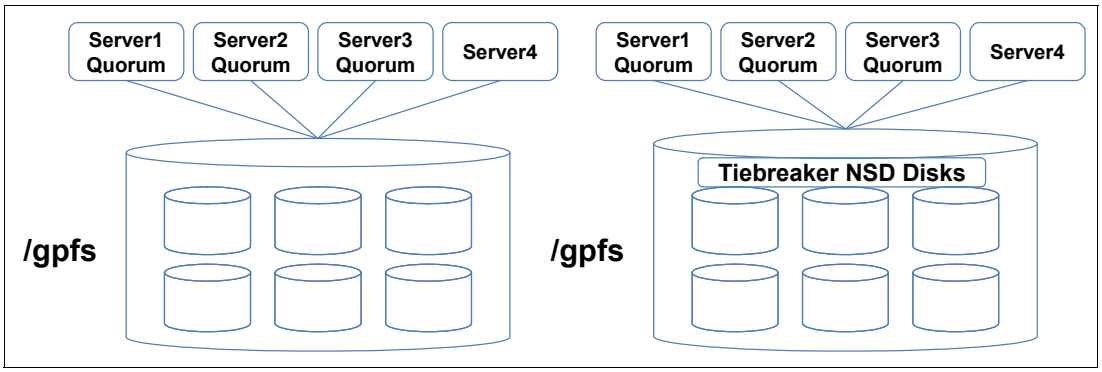


Figure 2-15 IBM Spectrum Scale node quorum and tiebreaker disks

The quorum must be maintained within the cluster for IBM Spectrum Scale to remain active. If the quorum semantics are broken, IBM Spectrum Scale performs recovery in an attempt to achieve quorum again.

IBM Spectrum Scale can use any one of the following methods for determining quorum:

- Node quorum: This algorithm is the default for IBM Spectrum Scale. Note the following information about node quorum:
  - Quorum is defined as one plus half of the explicitly defined quorum nodes in the Spectrum Scale cluster.
  - There are no default quorum nodes; you must specify which nodes have this role.
- Node quorum with tiebreaker disks: As an alternative to the node-based quorum, quorum based on tiebreaker disks allows the Spectrum Scale cluster to remain online with only one surviving node when it has access to the tiebreaker disks. To achieve this, add a tiebreaker disk to the quorum configuration.

  Note the following information about this algorithm:
  - Allows you to run with as little as one quorum node available if you have access to a majority of the quorum disks.
  - Up to eight quorum nodes are allowed in a cluster with quorum based on tiebreaker disks.
  - Enabling these disks starts by designating one or more nodes as quorum nodes. Then, define one to three disks as tiebreaker disks by using the **`tiebreakerDisks`** parameter in the **`mmchconfig`** command: **`mmchconfig tiebreakerDisks="Disk1 [,Disk2,Disk3]"`**.
  - You can designate any disk in the file system to be a tiebreaker or you can create a dedicated disk of type *descOnly*.

Note: Tiebreaker disk rules:

- Although you can have one, two, or three tiebreaker disks, a good practice is to use an odd number of tiebreaker disks.
- Among the quorum node groups that appear after an interconnect failure, only those having access to a majority of tiebreaker disks can be candidates to be the survivor group.
- Tiebreaker disks must be connected to all quorum nodes.

Cluster configuration repository

The Spectrum Scale cluster configuration data stored in the `/var/mmfs/gen/mmsdrfs` file is maintained on the nodes designated as the primary Spectrum Scale cluster configuration server and, if specified, the secondary Spectrum Scale cluster configuration server. When running Spectrum Scale administration commands, it is necessary that the Spectrum Scale cluster configuration data is accessible to the node running the command. Commands that update the `mmsdrfs` file require that both the primary and, if specified, the secondary Spectrum Scale cluster configuration server nodes are accessible. If one of the cluster configuration server nodes is inaccessible, you can designate a new primary or secondary cluster configuration server by using the **`mmchcluster`** command. Similarly, when the Spectrum Scale daemon starts, at least one of the two server nodes must be accessible.

IBM Spectrum Scale 4.1 uses by default Cluster Configuration Repository (CCR) to maintain the cluster configuration data, rather than using the traditional method based on designating a primary and a secondary configuration server at cluster creation time, also known as *server-based repository*. It uses all defined quorum nodes to maintain the Spectrum Scale cluster configuration data files. You do no longer have a designated primary and secondary configuration server role.

Note: IBM Spectrum Scale 4.1 uses CCR as default when running the **`mmcrcluster`** command, unless specified in command syntax the option '--ccr-disable' together with a primary configuration server, and optionally a secondary configuration server.

Using CCR has the advantage that full read/write access to the configuration data remains available when a majority of quorum nodes are accessible. For example, in a cluster with five quorum nodes, commands that update the `mmsdrfs` file continue to work normally, even if any two of the five quorum nodes have failed. In a two-node cluster with tiebreaker disks, it is still possible to run commands that change the `mmsdrfs` file if one of the two nodes has failed, as long as the surviving node has access to the tiebreaker disks.

Note: When the cluster has been configured with CCR, enabling or disabling the tiebreaker disk configuration can be done online, without requiring IBM Spectrum Scale to be stopped on nodes.

IBM Spectrum Scale 4.1 allows you to have both types of repositories:

- CCR (default): You enable CCR at the cluster creation time, by using **`mmcrcluster --ccr-enable`**, or without specifying any configuration repository type. You can configure it after creating the cluster in the standard mode (server-based repository). Enabling CCR can be done online. It does not require IBM Spectrum Scale to be stopped on nodes. However, returning from CCR model to the server-based repository configuration requires IBM Spectrum Scale to be stopped on all nodes.
- Server-based repository: This is the traditional configuration mode and the only one available in IBM Spectrum Scale versions before 4.1. If you are migrating from earlier versions to IBM Spectrum Scale 4.1, this configuration is adopted. You can modify the configuration to CCR by using the **`mmchcluster --ccr-enable`** command.

Note: You need to set the server-based repository type in a disaster recovery environment using Spectrum Scale replication with IBM Spectrum Scale 4.1. See more details in Chapter 7, "Backup and disaster recovery using IBM Spectrum Scale" on page 345. This restriction is going to be removed in future IBM Spectrum Scale releases.

See also several examples on how to manage the CCR configuration in 4.2.1, “Managing cluster repository” on page 187.

Limiting the nodes that can act as the cluster manager

The cluster manager is one of the nodes in the Spectrum Scale cluster, which is elected from the set of quorum nodes. It performs several tasks in the Spectrum Scale cluster:

- Monitors disk leases.
- Detects failures and manages recovery from node failure within the cluster. The cluster manager determines whether or not a quorum of nodes exists to allow the Spectrum Scale daemon to start and for file system usage to continue.
- Distributes certain configuration changes that must be known to nodes in remote clusters.
- Selects the file system manager node.
- Handles user ID (UID) mapping requests from remote cluster nodes.

Regardless of the quorum configuration used in the Spectrum Scale cluster (default node quorum or tiebreaker disk), a quorum node may become the cluster manager, even if that node has not been designated as a “manager” node. The roles of manager/client and quorum/nonquorum are assigned to the cluster nodes when creating the cluster (**mmcrcluster**) or adding nodes to the cluster (**mmaddnode**). They can also be changed by using the **mmchnode** command.

Starting with IBM Spectrum Scale 4.1, a special parameter named *clusterManagerSelection* has been introduced for limiting the nodes that can be elected as cluster managers.

The following values are allowed for this parameter:

- *quorumNode*: Any quorum node is eligible to become a cluster manager.
- *preferManager*: Choose the lowest-numbered node from the set of managers nodes (as shown by the **mmlscluster** command), if one is reachable (default in IBM Spectrum Scale 4.1).
- *enforceManager*: It restricts that only manager nodes can be selected, and non-manager nodes will not attempt to run the election. This option assumes that there are manager nodes in the cluster that are also assigned as quorum nodes.

You can change the *clusterManagerSelection* by using the **mmchconfig** command. The Spectrum Scale cluster services need to be stopped on all nodes for performing the change in configuration.

File system descriptor quorum

A *file system descriptor quorum* is initially written to every disk in the file system and is replicated on a subset of the disks as changes to the file system occur, such as the adding or deleting of disks.

Based on the number of failure groups and disks, IBM Spectrum Scale creates one to five replicas of the descriptor (this is the same file system descriptor that is created with the **mmcrfs** command):

- If at least five separate failure groups exist, five replicas are created. If at least five separate replicas exist, IBM Spectrum Scale can tolerate a loss of two of the five replicas.
- If at least three separate disks exist, three replicas are created. If at least three replicas exist, IBM Spectrum Scale can tolerate a loss of one of the three replicas.
- If only one or two disks exist, a replica is created on each disk. If fewer than three replicas exist, a loss of one replica might cause the descriptor to be inaccessible.

Figure 2-16 describes the file system descriptor with replication and storage pools. The diagram shows three tiebreakers for one Spectrum Scale cluster, and three descriptors for one Spectrum Scale file system. This configuration allows the loss of one failure group.

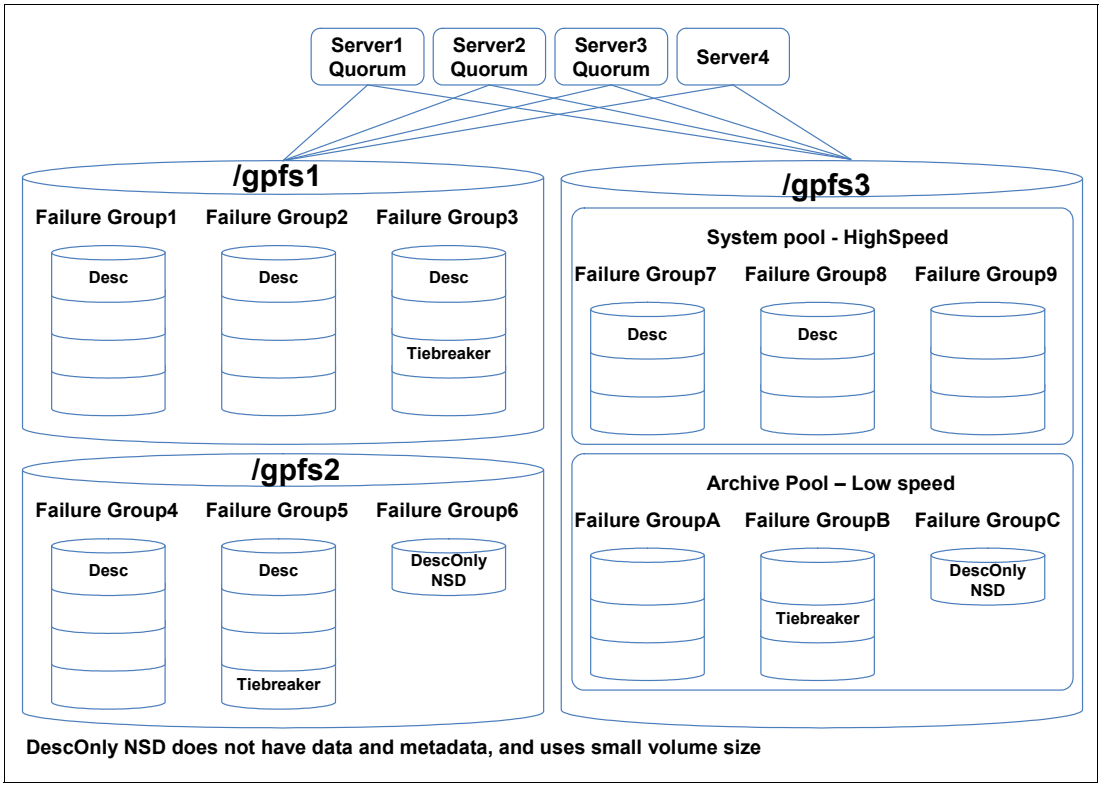


Figure 2-16 File system descriptor with replication and storage pool

After it decides how many replicas to create, IBM Spectrum Scale picks disks to hold the replicas, so that all replicas are in separate failure groups, if possible, to reduce the risk of multiple failures. In picking replica locations, the current state of the disks is considered. Stopped or suspended disks are avoided. Similarly, when a failed disk is brought back online, IBM Spectrum Scale might modify the subset to rebalance the file system descriptors across the failure groups. The disks that are used to hold the file system descriptor replicas can be seen by running the following command and looking for the string `desc` in the remarks column:

```
mmlsdisk fsname -L
```

The loss of all disks in a disk failure group might cause a majority of file system descriptors to become unavailable and inhibit further file system operations. For example, if your file system is backed up by three or more disks that are assigned to two separate disk failure groups, one of the failure groups will be assigned two of the file system descriptor replicas. The other failure group will be assigned only one replica. If all of the disks in the disk failure group that contains the two replicas become unavailable, the file system also becomes unavailable. To avoid this particular scenario, consider introducing a third disk failure group consisting of a single disk that is designated as a descOnly disk.

Note: A disk that is designated as a descOnly does not contain any file system data. This disk must have a minimum size of 128 MB.

NSD server and disk failure

The three most common reasons why data becomes unavailable are as follows:

- Disk failure
- Disk server failure with no redundancy (configure local storage)
- Failure of a path to the disk

When there is a disk failure in which IBM Spectrum Scale can no longer read or write to the disk, IBM Spectrum Scale discontinues the use of the disk until it returns to an available state. You can guard against loss of data availability from disk failure by using the following items:

- Hardware data protection as provided by a RAID
- The Spectrum Scale data and metadata replication features

Figure 2-17 describes the NSD server failover architecture.

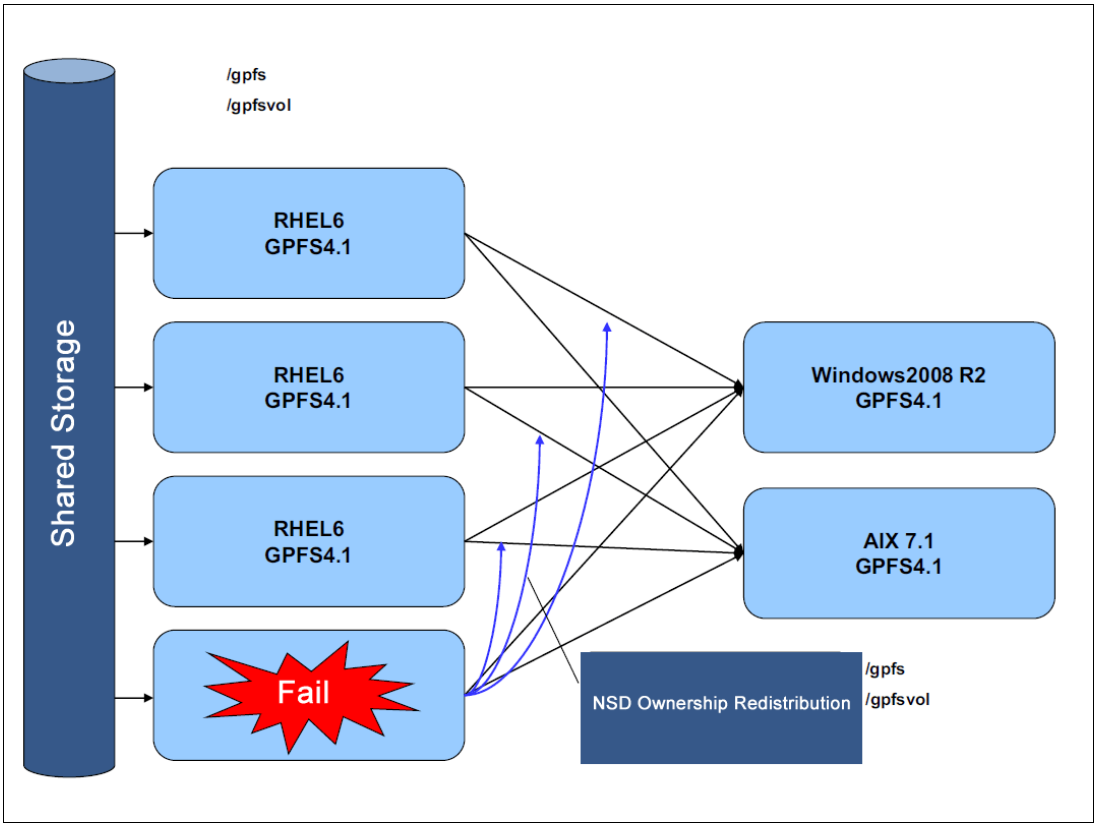


Figure 2-17 NSD server failure and rebalancing NSD volume ownership on the NSD server

Consider RAID as the first level of redundancy for your data and add Spectrum Scale replication if you want additional protection. We recommend having at least the metadata replicated. It takes a small amount of space, and provides more reliability to the file system.

When there is an NSD server failure in which a Spectrum Scale client can no longer contact the node that provides remote access to a disk, IBM Spectrum Scale discontinues the use of the disk. You can guard against loss of an NSD server availability by using common disk connectivity on multiple NSD server nodes and specifying multiple NSD servers for each common disk.

Note: If a path to a disk fails, IBM Spectrum Scale reports a disk failure and marks the disk as being down. To bring the disk back online, first follow the directions that are supplied by your storage vendor to determine and repair the failure. Then, the **mmchdisk** command needs to be run to bring any disks that have been marked down, back into a ready state.

2.6.2 Network Shared Disk creation considerations

This section explains considerations when you create a Network Shared Disk (NSD) on IBM Spectrum Scale.

- Considerations for creating an NSD
- Operating NSD server and client

NSD configuration guidance

You must prepare each physical disk that you intend to use with IBM Spectrum Scale by first defining it as an NSD by using the **mmcrnsd** command. NSDs can be created on the following types of physical disks:

- A block device defined in the `/dev` directory on UNIX, which needs to be associated with a disk device.
- An empty disk drive on Windows operating systems. The **mmcrnsd** commands accept Windows Basic disks or Unknown/Not Initialized disks.

The disk devices in the OS can have different names, depending on the platform and device driver used. Table 2-7 shows a list of currently supported devices in IBM Spectrum Scale.

Table 2-7 IBM Spectrum Scale known devices

Platform	Operating system device name	Spectrum Scale device type	Description
AIX	hdisk	hdisk	AIX hard disk.
	vpath	vpath	IBM Virtual path disk.
	hdiskpower	powerdisk	EMC powerpath disk.
	dlmfdrv	dlmfdrv	Hitachi dlm.
Linux	dm-	dmm	Device-Mapper Multipath.
	vpath	vpath	IBM virtual path disk.
	sd or hd	generic	General disk device having no unique failover or multipathing characteristic.
	emcpower	powerdisk	EMC power path disk.
Windows operating system	Device# (0-n)	gpt	Spectrum Scale partition on Windows system disk (number of partitions as shown by diskpart utility).

When the Spectrum Scale daemon starts on a node, it discovers the disks that are defined as NSDs by reading a disk descriptor that is written on each disk owned by IBM Spectrum Scale. On UNIX, NSD discovery is done by the Spectrum Scale shell script

/usr/lpp/mmfs/bin/mmdevdiscover, which generates a list of available disk devices that appear in the node's local /dev file system. To override or enhance NSD discovery, you can create a script and name it /var/mmfs/etc/nsddevices. The user-created nsddevices script, if it exists, is executed before the default discovery process. Refer to the nsddevices file in /usr/lpp/mmfs/samples for an example on how to create such a script. On Windows operating systems, NSDs have a GUID Partition Table (GPT) with a single Spectrum Scale partition. NSD discovery is done by scanning the system for a list of disks that contain a Spectrum Scale partition.

In IBM Spectrum Scale 4.1, a new NSD format is available called *NSDv2*. By using this format during the **mmcrnsd** command on a disk device, IBM Spectrum Scale on Windows operating systems and Linux reinitializes the disk that is used for NSD, so that it becomes a GPT disk with a single Spectrum Scale partition. IBM Spectrum Scale 4.1 can recognize also the NSDs from previous releases. The use of NSDv1(previous version of IBM Spectrum Scale 4.1) or NSDv2 is determined by the **minReleaseLevel** parameter, as shown by the **mmlsconfig** command. NSD data is stored in Spectrum Scale partitions, allowing other operating system components to recognize that the disks are used. The **mmdelnsd** command deletes the partition tables that are created by that **mmcrnsd** command.

As input, the **mmcrnsd** command expects a file, DescFile, that contains a disk descriptor, one stanza for each of the disks to be processed.

Note: Starting with GPFS 3.5, the file format for the nsd disks has been changed. The old format is still accepted in GPFS 3.5 and IBM Spectrum Scale 4.1. For compatibility reasons, however, its use is discouraged.

Disk descriptors have the following format:

```
%nsd: device=DiskName
  nsd=NsdName
  servers=ServerList
  usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
  failureGroup=FailureGroup
  pool=StoragePool
```

Disk descriptors can be explained as follows:

- device=DiskName
  - On UNIX, this parameter is the block device name that appears in /dev for the disk you want to define as an NSD. Examples of disks that are accessible through a block device are directly attached disks. The Spectrum Scale disk discovery looks for common device types, /dev/sd on Linux for example. If a **ServerList** parameter is specified, DiskName must be the /dev name for the disk device on the first NSD server node that is defined in the server list. The reason is because, when the **mmcrnsd** command is executed, the NSD descriptors are written by the first node in the **ServerList** parameter.
  - On Windows operating system, this parameter is the disk number of the disk that you want to define as an NSD. Disk numbers appear in the Windows Disk Management console and the DISKPART command-line utility. If a server node is specified, DiskName must be the disk number from the first NSD server node defined in the server list.

  All device names under the same operating system point to the same LUN devices, and configuration of mixed operating systems with the same LUN is not supported.

- `servers=ServerList`

  This parameter is a comma-separated list of NSD server nodes and has the following form:

  `server1[,server2,...,server8]`

  You can specify up to eight NSD servers in this list. A Spectrum Scale node accessing the disk through an NSD server uses the first server in the list with which it can communicate. If the first server is not available, the node uses the next available server in the list. If you do not define a server list, IBM Spectrum Scale assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other Spectrum Scale clusters, you must specify a server list.

- `nsd=nsdName`

  This parameter specifies the name of the NSD to be created. This name must not already be used as another Spectrum Scale disk name, and it must not begin with the reserved string 'gpfs'.

  If a wanted name is not specified, the NSD is assigned a name according to the following convention:

  `gpfsNNnsd`

  In this convention, NN is a unique nonnegative integer that is not used in any prior NSD.

  Note: This name can contain only the following characters:

  - A - Z
  - a - z
  - 0 - 9
  - Underscore (_) character

  All other characters are not valid.

- `usage=DiskUsage`

  Use this parameter to specify disk usage or accept the default. This parameter is used at file system creation, so it is ignored by the **mmcrnsd** command and is passed unchanged to the output descriptor file produced by the **mmcrnsd** command. Possible values are as follows:

  – dataAndMetadata: Indicates that the disk contains both data and metadata. This value is the default for the system storage pool.

  – dataOnly: Indicates that the disk contains data and does not contain metadata. This value is the default and only allowed value for all storage pools other than the system pool.

  – metadataOnly: Indicates that the disk contains metadata and does not contain data. Only NSDs in the system pool can be designated metadataOnly.

  – descOnly: Indicates that the disk contains no data and no metadata. Such a disk is used solely to keep a copy of the file system descriptor. This disk usage is most commonly used as a third failure group in certain disaster recovery configurations.

  – localCache: Indicates that nsd is used for local read-only cache. The local read-only cache disk is expected to be a solid-state disk (SSD) accessible via SCSI. The device as defined as a standard NSD by **mmcrnsd**, but the DiskUsage is set to localCache. The NSD must have a primary server and is not allowed to have other servers. The primary server must be the node where the physical local read-only cache device is installed.

The device is not exported to other nodes in the cluster. The storage pool and failure group defined for NSD are ignored and should be set to null. The **mmcrnsd** command creates a unique NSD ID and name for the device and updates sector 2 to indicate that it is an NSD.

Note: Local read-only cache is a new feature introduced in IBM Spectrum Scale 4.1. It is available currently only on Linux x86 systems.

- `failureGroup=FailureGroup number`

  This parameter is a number that identifies the failure group to which this disk belongs. A failure group identifier can be a simple integer or a topology vector consisting of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk. If you do not specify an NSD server, the value defaults to -1. IBM Spectrum Scale uses this information during data and metadata placement to ensure that no two replicas of the same block are written in such a way as to become unavailable because of a single failure. All disks that are attached to the same NSD server or adapter must be placed in the same failure group. This field is ignored and passed unchanged to the output descriptor file written by the **mmcrnsd** command.

- `pool=StoragePool`

  This parameter specifies the name of the storage pool to which the NSD is assigned. The following guidelines exist for providing storage pool names:

  – Must be unique within a file system, but not across file systems.
  – Must not be larger than 255 alphanumeric characters.
  – Are case-sensitive.

  For example, “MYpool” and “myPool” are distinct storage pools. If this name is not provided, the default is system. Only the system pool can contain metadataOnly, dataAndMetadata, or descOnly disks.

  Note: Only disks in system pool can be used as dataAndMetadata, metadataOnly, or descOnly. Disks in other pools use only the dataOnly usage type.

  See more details about storage pool options in 2.6.3, “Planning the IBM Spectrum Scale file system” on page 85.

NSD server considerations

If you plan to use NSD servers to remotely serve disk data to other nodes, as opposed to having disks SAN-attached to all nodes, be sure to consider the total computing and I/O load on these nodes, as follows:

- Determine whether your NSD servers should be dedicated servers or whether you will also be using them to run applications. If you will have non-dedicated servers, consider running fewer time-critical applications on these nodes. If you have runtime-critical applications on an NSD server, servicing disk requests from other nodes might conflict with the demands of these applications.
- Because the special functions of the file system manager consume extra processing time, if possible, avoid using an NSD server as the file system manager. The NSD server consumes both memory and processor cycles that can affect the operation of the file system manager.
- The actual processing capability required for NSD service is a function of the application I/O access patterns, the type of node, the type of disk, and the disk connection. You can

later run the **iostat** command on the server to determine how much of a load your access pattern places on an NSD server.

- Provide sufficient disks and adapters on the system to yield the required I/O bandwidth. Dedicated NSD servers must have sufficient disks and adapters to drive the I/O load you expect them to handle.
- Know approximately how much storage capacity you need for your data.

Consider what you want as the default behavior for switching between local access and NSD server access if there is a failure. To set this configuration, use the **-o useNSDserver** file system mount option of the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands to control behavior:

- To specify the disk discovery behavior.
- To limit or eliminate switching from either the local access to NSD server access or NSD server access to local access.
- To eliminate performance degradation from unwanted switching to network access.

Consider specifying how long to wait for an NSD server to come online before allowing a file system mount to fail because the server is not available.

The **mmchconfig** command has the following options:

- `nsdServerWaitTimeForMount`

  When a node is trying to mount a file system, which has disks that depend on NSD servers, this option specifies the number of seconds to wait for those servers to be up. If a server recovery is taking place, the wait time you specify with this option starts after recovery completes.

- `nsdServerWaitTimeWindowOnMount`

  This option specifies a window of time (in seconds) during which a mount can wait for NSD servers, as described for the `nsdServerWaitTimeForMount` option. The window begins when quorum is established (at cluster start or subsequently), or at the last known failure times of the NSD servers that are required to perform the mount.

Notes:

- When a node rejoins a cluster, it resets all the failure times that it knew about within that cluster.
- Because a node that rejoins a cluster resets its failure times within that cluster, the NSD server failure times are also reset.
- When a node attempts to mount a file system, IBM Spectrum Scale checks the cluster formation criteria first. If that check falls outside the window, it then checks for NSD server fail times being in the window.

Figure 2-18 on page 83 shows the NSD client performance, where the I/O balanced configuration is important. Each server must have its own NSD.

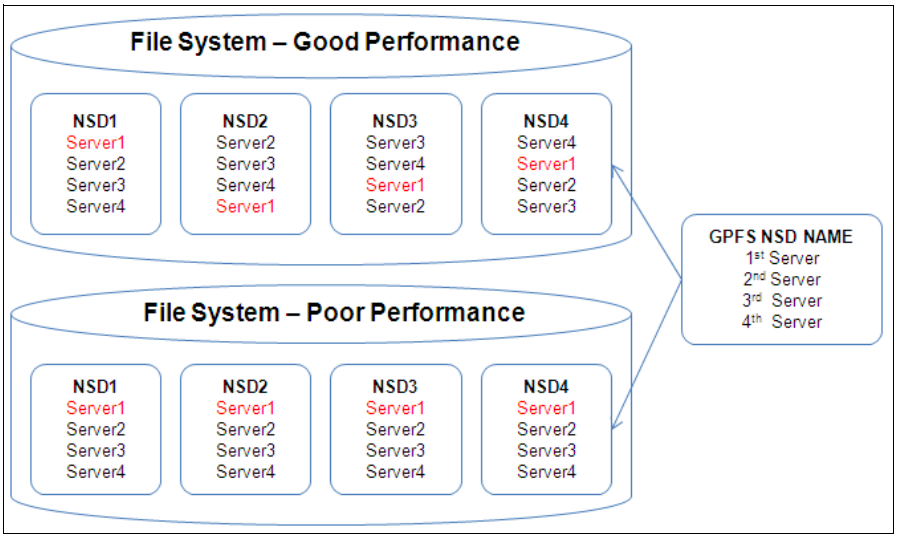


Figure 2-18 Server assignments on all NSDs

NSD server and client access operation

Usually, this configuration is based on shared storage and local storage. On the Spectrum Scale NSD client-side view, shared storage and local storage have TCP connections with all Spectrum Scale servers, and for writing a file, the work is divided across all Spectrum Scale servers. For example, the sample file is 300 MB and there are three Spectrum Scale servers. In this case, a 100-MB file is written to its own storage device for each.

Figure 2-19 shows how the operation goes on the Spectrum Scale NSD client when writing a file.

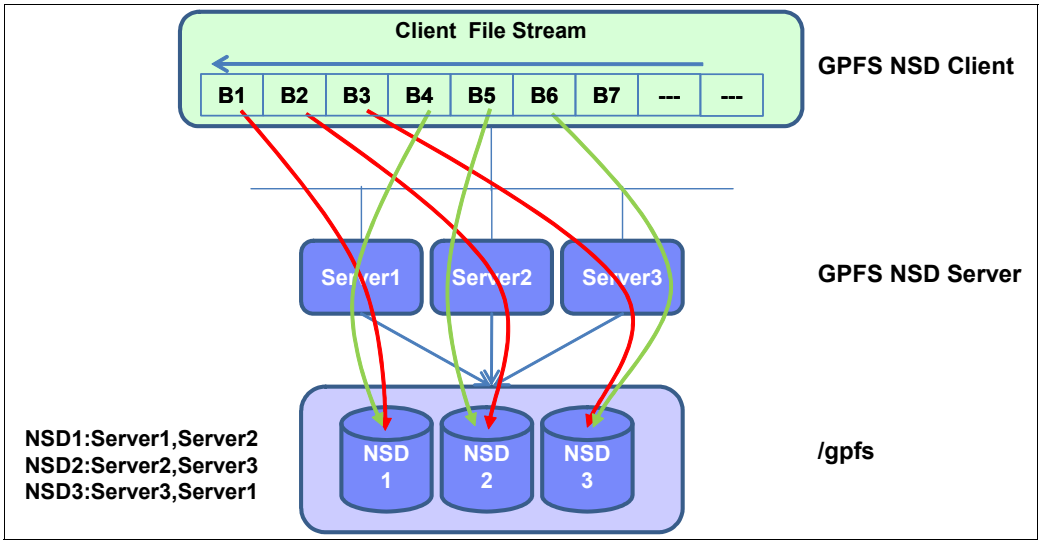


Figure 2-19 NSD client and server I/O stream

For Figure 2-19, the following observations can be mentioned:

- Spectrum Scale NSD client: The Spectrum Scale volume access is through the Spectrum Scale NSD client's own network device.
- Spectrum Scale NSD server: This volume is exported by IBM Spectrum Scale.

Direct NSD access operation

This configuration has direct access to all the LUNs in the storage system. Figure 2-20 describes how the operation on the Spectrum Scale server happens when writing a file.

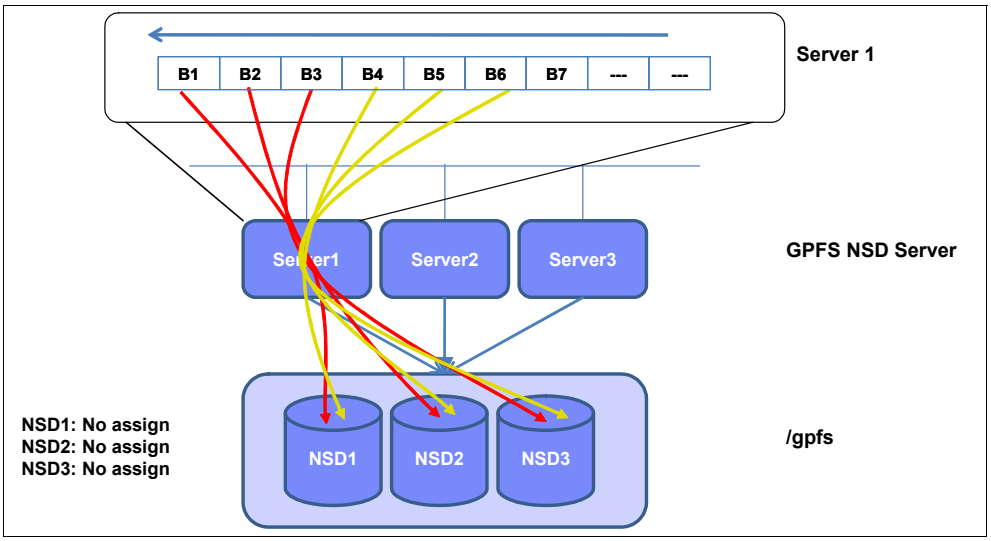


Figure 2-20 No NSD configuration I/O stream

Disk access method: local versus shared disks

The Spectrum Scale NSD configuration requires a local disk or shared disk. The following configurations can be used for the disk access method:

- Multi-platform: This configuration is shared disk on the same operating system and heterogeneous environment on a single Spectrum Scale cluster.
- Shared storage: This design does not require an assigned NSD server, but cannot mix operating systems. Although it is an expensive solution, it provides more high availability, reliability, and high performance.
- Local storage: This configuration offers good performance. Performance depends on the network switch. When using this configuration, the NSD server must be defined on local disk. If the node fails on this cluster, the file system is shut down. So, remember to take advantage of the local disk RAID configuration and the Spectrum Scale replication with a local disk tiebreaker configuration.

Figure 2-21 shows a diagram of three Spectrum Scale cluster configurations:

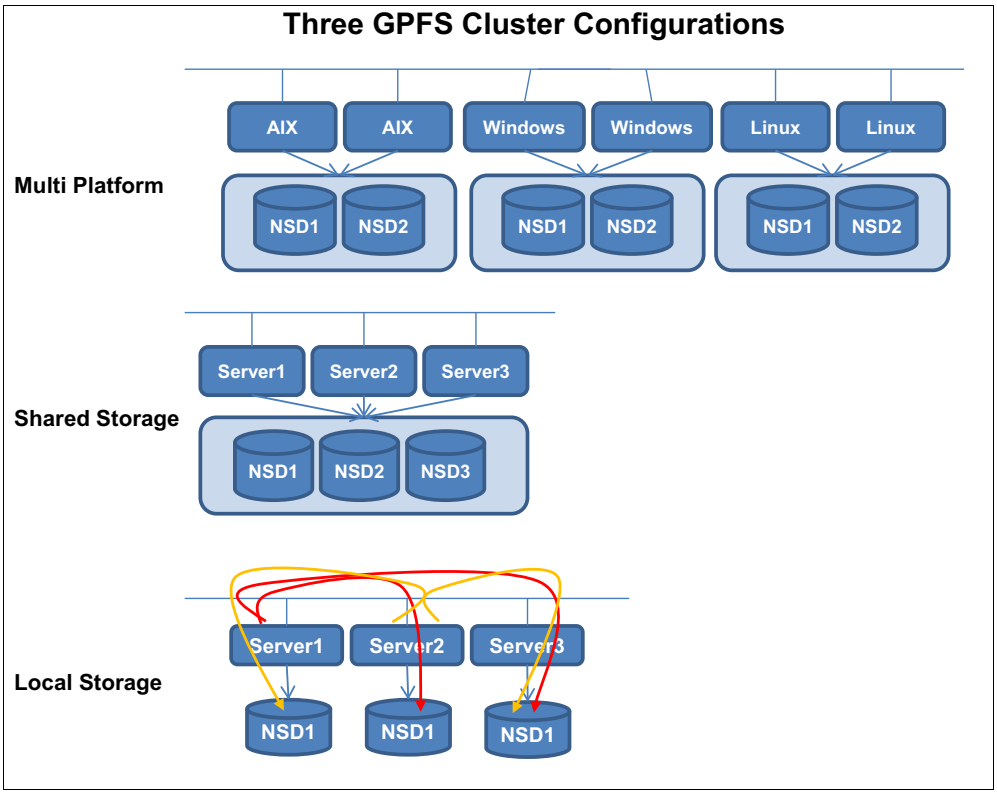


Figure 2-21 Three Spectrum Scale cluster configurations

Single node cluster

The Spectrum Scale cluster can be configured with 1 - 8192 nodes. If you want more high availability functions with your Spectrum Scale cluster, configure at least a three-node cluster or set up two nodes with tiebreaker disks. A single node Spectrum Scale cluster does not provide high availability functions.

For the particular case of using IBM Spectrum Scale in a disaster recovery configuration based on Spectrum Scale replication, a third site is active. It consists of a single node and a single disk that is used as a tiebreaker for Spectrum Scale quorum. If there is a catastrophic hardware failure that disables the operation of an entire site, and assuming the tiebreaker site remains operational, file system services fail over to the remaining subset of the cluster and continue serving the data using the replica of the file system that survived the disaster.

2.6.3 Planning the IBM Spectrum Scale file system

When creating a file system there are two types of parameters: Those that can be changed dynamically and those that cannot.

Note: A key parameter when creating the Spectrum Scale file system is the block size. After it is set, the only way to change the block size is to re-create the file system. Therefore, be sure to test the block size on a development system before production deployment.

IBM Spectrum Scale 4.1 supports block sizes in the range of 64 KB - 16 MB on 2^n boundaries. The default block size is 256 KB. The block size of a file system is determined at

creation using the **-B** parameter to the **mmcrfs** command. In addition to using the **-B** parameter to create a file system with a block size greater than 1 MB, you must increase the value of the **maxblocksize** parameter. The default value of maxblocksize is 1 MB and the allowable range is 64 KB - 16 MB.

You can change the **maxblocksize** parameter using: **mmchconfig maxblocksize=<newvalue>**. IBM Spectrum Scale must be stopped on all cluster nodes when changing the maxblock value.

How do you choose a block size for your file system? It is best to test the impact of various blocksize settings with your application. If you cannot test various values of block sizes or you are looking only for a starting point, see Table 2-8, which provides basic guidance for what file system block sizes might be appropriate for various types of applications.

Table 2-8 Block size configuration guide, by application

I/O type	Application examples	Block size
Large sequential I/O	Scientific computing, digital media	1 ~ 4 MB
Relational database	DB2, Oracle	512 KB
Small I/O sequential	General file service, file-based analytics	256 KB
Small files	Email, web application servers	64 KB

The file system block size setting is the amount of data written to each disk in a file system before moving on to the next disk. The following characteristics are important to understand when you consider the appropriate value of block size:

- The block size is the largest size I/O that IBM Spectrum Scale can issue to the underlying device.
- A subblock is 1/32 of the block size. This value is the smallest allocation to a single file.
- Common sector size is 512 bytes. The newer disks have been developed with 4 K sector size. This number is the smallest I/O request size that IBM Spectrum Scale issues to the underlying device.

This information means that, for example, if you use a block size of 1 MB, each file uses at least 32 KB (1024 KB / 32 = 32 KB).

What if you do not know your application I/O profile? Often, you do not have good information about the nature of the I/O profile, or the applications are so diverse that optimizing for one or the other is difficult. Generally, two approaches to designing for this type of situation, *separation* or *compromise* are available:

- Separation

  In this model, you create two file systems, one with a large file system block size for sequential applications and one with a smaller block size for small file applications. You can gain benefits from having file systems of two different block sizes even on a single type of storage. You can also use separate types of storage for each file system to further optimize to the workload.

  In either case, the idea is that you provide two file systems to your users, for scratch space on a compute cluster for example. The users can run tests themselves by pointing the application to one file system or another to determine, by direct testing, which is best for their workload. In this situation, you may have one file system optimized for sequential I/O with a 1 MB block size, and one for more random workloads at the 256 KB block size.

- Compromise

  In this situation, you either do not have sufficient information about workloads (that is, users will not think about I/O performance) or enough storage for multiple file systems. In this case, a general suggestion is use a block size of 256 KB or 512 KB, depending on the general workloads and storage model used. With a 256 KB block size, you can still have good sequential performance (although not necessarily peak marketing numbers). You can also have good performance and space utilization with small files (256 KB has minimum allocation of 8 KB to a file). This configuration is good for multi-purpose research workloads where the application developers are focusing on their algorithms more than I/O optimization.

Block allocation map

IBM Spectrum Scale has two methods of allocating space in a system (cluster and scatter). You can use the **-j** option when creating a file system. This block allocation map type cannot be changed after the file system is created. Usually, IBM Spectrum Scale first uses a round-robin algorithm to spread the data across all NSDs in the file systems.

Notes:

- After the disk is selected, the location of the data block on the disk is determined by the block allocation map type.
- The block allocation map mode (scatter/cluster) has nothing to do with the way data is spread across NSDs. This mode controls how data is distributed on a given NSD after it is selected by round robin.

Table 2-9 summarizes the block allocation map of each of option.

Table 2-9 Comparing types of block allocation

Type	Cluster	Scatter
Node configure	Eight or fewer nodes	More than eight nodes
Pattern	Sequential	Random
Benefit	Block allocation diminish	Consistent file system

Storage pool options

IBM Spectrum Scale storage pools are used to provide capability to partition a file system based on underlying storage characteristics, particularly used in Information LifeCycle Management (ILM) configurations. Besides the ILM use, the storage pools are also used for IBM Spectrum Scale FPO configurations.

In GPFS 3.5 and IBM Spectrum Scale 4.1, you can define specific file system options at the storage pool level. The options can be provided in the stanza file used as input to the **mmcrfs** command. Following is the format of the pool options stanza:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

The parameters have the following descriptions:

- pool=StoragePoolName, is the name of the storage pool.
- blockSize=BlockSize, specifies the block size of the disks in the storage pool.

> **Note:** This parameter is intended for differentiating between block sizes of data and metadata in file systems using separate pools for them, so file systems I/O operations can be improved. The metadata pool can be only the system pool. In case of data, if there are multiple pools, all of them must use the same block size.

- usage={dataOnly | metadataOnly | dataAndMetadata}. Specifies the type of data to be stored in the storage pool.
- layoutMap={scatter | cluster}. The block allocation map type cannot be changed after the storage pool has been created.
- allowWriteAffinity={yes | no}. Indicates whether the IBM Spectrum Scale File Placement Optimizer (FPO) feature is to be enabled for the storage pool.
- writeAffinityDepth={0 | 1 | 2}. Specifies the allocation policy to be used by the node writing the data. It is also used for FPO-enabled pools.
- blockGroupFactor=BlockGroupFactor. Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works on FPO enabled pools, where --allow-write-affinity is set for the data pool.

  For the options related to the FPO configurations, see more details in 2.6.4, “Planning for IBM Spectrum Scale FPO” on page 93.

Replication parameters

The metadata and data replication are set at the file system level and apply to all files. They are initially set for the file system when issuing the **mmcrfs** command. They can be changed for an existing file system by using the **mmchfs** command. When the replication parameters are changed, files created after the change are affected. To apply the new replication values to existing files in a file system, issue the **mmrestripefs -r** command.

> **Note:** If you want to configure the replication function, before creating a file system, you must have a failure group number when creating the NSD and the data disk in each storage pool. Plan carefully for the failure groups according to your storage layout and node attachment. IBM Spectrum Scale cannot validate the failure group proper assignment to NSDs. There is no relationship between the failure group numbers. For the FPO case, the failure groups have a special format. See 2.6.4, “Planning for IBM Spectrum Scale FPO” on page 93.

The following list provides information about the metadata and data replicas:

- Default metadata replicas

  The default number of copies of metadata for all files in the file system can be specified at file system creation by using the **-m** option on the **mmcrfs** command or changed later by using the **-m** option on the **mmchfs** command. This value must be equal to or less than the **MaxMetadataReplicas** parameter, and cannot exceed the number of failure groups with disks that can store metadata. The allowable values are 1 or 2, or 3 with GPFS 3.5.0.7 or later, with a default of 1.

- Maximum metadata replicas

  The maximum number of copies of metadata for all files in the file system can be specified at file system creation by using the **-M** option on the **mmcrfs** command. The default is 2.

The allowable values are 1 or 2, or 3 with GPFS 3.5.0.7 or later, but it cannot be lower than the value of the **DefaultMetadataReplicas** parameter. This value cannot be changed.

- Default data replicas

  The default replication factor for data blocks can be specified at file system creation by using the **-r** option on the **mmcrfs** command or changed later by using the **-r** option on the **mmchfs** command. This value must be equal to or less than the **MaxDataReplicas** parameter, and the value cannot exceed the number of failure groups with disks that can store data. The allowable values are 1, 2, or 3 with GPFS 3.5.0.7 or later, with a default of 1.

  If you want to change the data replication factor for the entire file system, the data disk in each storage pool must have a number of failure groups equal to or greater than the replication factor. For example, a failure error message is issued if you try to change the replication factor for a file system to two, when the storage pool has only one failure group.

- Maximum data replicas

  The maximum number of copies of data blocks for a file can be specified at file system creation by using the **-r** option on the **mmcrfs** command. The default is 2. The allowable values are 1, 2, or, 3 with GPFS 3.5.0.7 or later, but cannot be lower than the value of **DefaultDataReplicas**. This value cannot be changed.

Figure 2-22 provides an example of replication using three failure groups. The data and metadata is replicated between the three failure groups. The shaded gray is the second image block and the white color is the original image block. In this example case, if a failure occurs on failure group2 (Physical Storage Box or LUN Group), the Spectrum Scale file system operations continue with the remaining failure groups.

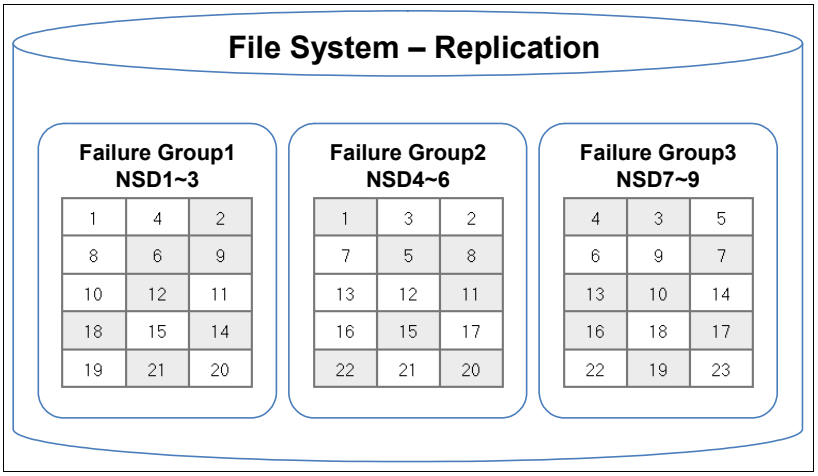


Figure 2-22 Spectrum Scale replication block diagram

Changing file replication attributes

Use the **mmchattr** command to change the replication attributes for one or more files. You can increase data and metadata replication only as high as the maximum data and maximum metadata replication factors for that file system. You cannot change the maximum data and maximum metadata replication factors after the file system has been created.

Storage pool with ILM

IBM Spectrum Scale provides storage management based on the definition and use of storage pools, policies, and file sets.

Storage pools

As you plan how to configure your storage, consider the following factors:

- Improved price-performance by matching the cost of storage to the value of the data.
- Improved performance by reducing the following items:
  - Contention for premium storage
  - Effect of slower devices
- Improved reliability by providing for the following items:
  - Replication based on need
  - Better failure containment

Policies

Files are assigned to a storage pool based on defined *policies*:

- Placement policies:

  Placing files in a specific storage pool when the files are created
- File management policies
  - Migrating files from one storage pool to another
  - Deleting files based on file characteristics
  - Changing the replication status of files

File sets

File sets provide a method for partitioning a file system and allow administrative operations at a finer granularity than the entire file system.

File sets allow you to do the following tasks:

- Define data block and inode quotas at the file set level
- Apply policy rules to specific file sets
- A fileset snapshot can be created to preserve the contents of a single independent file set plus all dependent file sets that share inode space

IBM Spectrum Scale supports independent and dependent filesets. An independent fileset is a fileset with its own inode space. An inode space is a collection of inode number ranges reserved for an independent fileset. An inode space enables more efficient per-fileset functions, such as fileset snapshots. A dependent fileset shares the inode space of an existing, independent fileset. Files that are created in a dependent fileset are assigned inodes in the same collection of inode number ranges that were reserved for the independent fileset from which it was created.

Snapshot

A *snapshot* of an entire Spectrum Scale file system or of an independent fileset can be created to preserve the contents of the file system at a single point in time. Snapshots can also be used with a lower granularity at the fileset level. The storage overhead for maintaining a snapshot is keeping a copy of data blocks that would otherwise be changed or deleted after the time of the snapshot.

Snapshots are read-only; changes can be made only to the active (that is, normal, non-snapshot) files and directories.

The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time that the snapshot was created. Snapshots also provide an online backup capability that allows easy recovery

from common problems such as accidental deletion of a file, and comparison with older versions of a file.

Notes:

- Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For information about protection against media failures, see “Recoverability considerations” in *General Parallel File System Concepts, Planning, and Installation Guide Version 4 Release 1*, GA76-0441-00.
- A snapshot of a file creates a new file that captures the user data and user attributes from the original. The snapshot file is independent from the original file. For DMAPI-managed file systems, the snapshot of a file is not automatically managed by DMAPI, regardless of the state of the original file. The DMAPI attributes from the original file are not inherited by the snapshot. For more information about DMAPI restrictions for IBM Spectrum Scale, see the *General Parallel File System Data Management API Guide Version 4 Release 1*, GA76-0441-00.

Exporting a file system using NFS

This section lists aspects to consider when you export a Spectrum Scale file system to NFS. The operating system being used and the version of NFS might require special handling or considerations:

- Linux export considerations:

  For Linux nodes only, issue the **`exportfs -ra`** command to initiate a reread of the `/etc/exports` file. Starting with Linux kernel version 2.6, an fsid value must be specified for each Spectrum Scale file system that is exported on NFS. For example, the format of the entry in the `/etc/exports` file for the `/gpfs1` Spectrum Scale file system to an external system *mach1*, might look as follows:

  ```
  /gpfs1 mach1(rw,fsid=745)
  ```

  Configuring the directories for export with NFSv4 differs slightly from the previous NFS versions. To configure the directories, do the following:

  a. Define the root of the overall exported file system (also referred to as the *pseudo root file system*) and the pseudo file system tree. For example, to define `/export` as the pseudo root and export the `/gpfs/dir1` directory, which is not below `/export`, run:

     ```
     mkdir –m 777 /export /export/dir1
     mount --bind /gpfs/dir1 /export/dir1
     ```

     In this example, `/gpfs/dir1` is bound to a new name under the pseudo root using the **`bind`** option of the **`mount`** command. This bind mount point should be explicitly unmounted after IBM Spectrum Scale is stopped and bind-mounted again after IBM Spectrum Scale is started. To unmount, use the **`umount`** command. For the preceding example, run the following command:

     ```
     umount /export/dir1
     ```

  b. Edit the `/etc/exports` file. There must be one line for the pseudo root with fsid=0. For the preceding example:

     ```
     /export cluster1(rw,fsid=0)
     /export/dir1 cluster1(rw,fsid=745)
     ```

- AIX export considerations

  AIX does not allow a file system to be exported by NFS V4 unless it supports NFS V4 ACLs.

- NFS export considerations for versions before NFS V4

  For NFS exported file systems, the version of NFS you are running with might have an impact on the number of inodes you need to cache, as set by both the maxStatCache and maxFilesToCache parameters on the **mmchconfig** command.

  The implementation of the **ls** command differs from NFS V2 to NFS V3. The performance of the **ls** command in NFS V3 in part depends on the caching ability of the underlying file system. Setting the cache large enough prevents rereading inodes to complete an **ls** command, but puts more of a CPU load on the token manager.

  Also, the clocks of all nodes in your Spectrum Scale cluster must be synchronized. If this is not done, NFS access to the data, and other Spectrum Scale file system operations, might be disrupted.

- NFS V4 export considerations

  For information about NFS V4, see the information at the NFSv4 General Information and References for the NFSv4 protocol website:

  https://datatracker.ietf.org/wg/nfsv4/documents

  To export a Spectrum Scale file system using NFS V4, there are two file system parameters that must be set (the attributes can be queried using the **mmlsfs** command, and set using the **mmcrfs**/**mmchfs** commands):

  – The '*-D nfs4*' flag is required. Conventional NFS access would not be blocked by concurrent file system reads or writes (this is the POSIX semantic). NFS V4 however, not only allows for its requests to block if conflicting activity is happening, it insists on it. Since this is an NFS V4 specific requirement, it must be set before exporting a file system.

  – The '*-k nfs4' or '-k all*' flag is required. Initially, a file system has the '-k posix' setting, and only traditional GPFS ACLs are allowed. To export a file system using NFS V4, NFS V4 ACLs must be enabled. Since NFS V4 ACLs are vastly different and affect several characteristics of the file system objects (directories and individual files), they must be explicitly enabled. This is done either exclusively, by specifying '-k nfs4', or by allowing all ACL types to be stored.

  Note: Setting ACLs to NFSV4 is also required when exporting the file system to Samba clients, or the Spectrum Scale file system is mounted on Windows operating system hosts.

Clustered NFS

IBM Spectrum Scale currently supports Clustered NFS (cNFS) only with Linux (X86_64 or ppc64) SLES or RHEL distributions supported by the version of IBM Spectrum Scale used in your environment. The following considerations apply:

- cNFS V4 is only supported by IBM Spectrum Scale V4.1.
- cNFS over IPV6 is only supported by IBM Spectrum Scale V4.1.
- All the nodes in the same group should use similar hardware and software running on them, and the configuration of IBM Spectrum Scale, NFS, network should be identical on nodes.
- cNFS is not supported on the Debian or Ubuntu distributions.
- cNFS is not able to export a remotely mounted file system.
- cNFS lock failover and failback has limitations on RHEL 6.x due to Linux kernel issues.

- For NFS v3 exclusive byte-range locking to work properly the following clients are required:
  - x86-64 with SLES 10 SP2 or later, SLES 11, RHEL 5.4 or later, and RHEL 6
  - ppc64 with SLES 11, RHEL 5.4 or later, and RHEL 6
- Kernel patches are required for distributions before SLES 10 SP2 and RHEL 5.2 (no kernel patches are required for SLES 11 or later, and RHEL5.4, or later). For more information, see IBM Spectrum Scale FAQs:

  http://ibm.co/1IKO6PN

2.6.4 Planning for IBM Spectrum Scale FPO

IBM Spectrum Scale FPO is an implementation of shared-nothing architecture that enables each node to operate independently, reducing the impact of failure events across multiple nodes using local data. Usually the FPO feature is used for big data applications such as IBM InfoSphere BigInsights or IBM Platform Symphony®. By storing your data using FPO, you are freed from some architectural restrictions related to Hadoop Distributed File System (HDFS). You can also take advantage of the IBM Spectrum Scale FPO features as a multipurpose file system to get additional management flexibility. For more details about IBM Spectrum Scale advantages with BigInsights, see a comparison of HDFS and IBM Spectrum Scale features in the IBM Knowledge Center:

http://ibm.co/1F6FgI8

You can manage your data using existing toolsets and processes along with a wide range of enterprise-class data management functions offered by IBM Spectrum Scale, including:

- Full POSIX compliance
- Snapshot support for point-in-time data capture
- Simplified capacity management by using IBM Spectrum Scale for all storage needs
- Policy-based information lifecycle management capabilities to manage petabytes of data
- Control over placement of each replica at file-level granularity
- Infrastructure to manage multi-tenant Hadoop clusters based on service-level agreements (SLAs)
- Simplified administration and automated recovery

Starting GPFS 3.5.0.11, FPO is formally supported to extend the core IBM Spectrum Scale architecture, providing greater control and flexibility to leverage data location, reduce hardware costs and improve I/O performance. It is important to understand key concepts and terms before you configure FPO:

- **NSDs and disks**: In a Spectrum Scale cluster, a physical disk and NSD can have a 1:1 mapping. In this case, each node in the cluster is an NSD server providing access to its disks for the rest of the cluster. In FPO deployments, it is important to correctly identify the disk types before creating the storage pools that are used for NSD definition and separate high-speed devices from traditional disks so the FPO can correctly move the data to the appropriate set of devices. In IBM Spectrum Scale, the use of the term “disk” is synonymous to NSD unless specified otherwise, mostly for historical reasons.
- **Failure groups**: The failure group definition allows locality information of the cluster nodes and the associated disks enable intelligent decision making for data block placements. Failure group is declared as an attribute of the disk when defining disks (NSDs).
- **Metadata and data placement**: IBM Spectrum Scale allows for separation of storage used for metadata and data blocks since they might have different requirements for availability, access, and performance. Metadata includes file metadata (file size, name, owner, last modified date); internal data structures such as quota information allocation maps, file system configuration and log files; and user-specified metadata such as inode,

directory blocks, extended attributes, and access control lists. The data includes data contents of a file. Metadata is vital to file system availability and should be stored on the most reliable storage media. The file data can be placed on the storage media based on the importance of the data. IBM Spectrum Scale provides rich policies for initial placement and subsequent movement of data between storage pools. IBM Spectrum Scale places metadata in a storage pool named *system*.

For better resiliency, distributing metadata across all available nodes is not recommended in an FPO environment. Having the metadata on too many nodes reduces the availability of the system because the probability of three nodes failing at the same time increases with the number of nodes. Therefore, it is recommended that you limit the number of nodes with metadata disks to one or two nodes per rack.

- **Replication**: IBM Spectrum Scale supports replication for failure containment and disaster recovery. Replication can incur costs in terms of disk capacity and performance, and therefore should be used carefully. IBM Spectrum Scale supports replication of both metadata and data independently and provides fine-grained control of the replication. You can choose to replicate data for a single file, a set of files, or the entire file system. Replication is an attribute of a file and can be managed individually or automated through policies. If replication is enabled, the target storage pool for the data must have at least two failure groups defined within the storage pool. Replication means that a copy of each block of data exists in at least two failure groups for two copies, and three failure groups for three copies. IBM Spectrum Scale supports up to three replicas for both metadata and data. The default replication factor is 1, meaning no replicas are created.

  In many IBM Spectrum Scale deployments, data protection is handled by the underlying RAID storage subsystem. In FPO environments, it is more common to rely on software replication for data protection.

- **Write-affinity depth (WAD)** Introduced for FPO, WAD is a policy for directing writes. It indicates that the node writing the data directs the write to disks on its own node for the first copy and to the disks on other nodes for the second and third copies (if specified). The policy allows the application to control placement of replicas within the cluster to optimize for typical access patterns. Write affinity is specified by a depth that indicates the number of replicas to be kept closer to the node ingesting data. This attribute can be specified at the storage pool or individual file level. The possible values are 0, 1, and 2. A write-affinity depth of 0 indicates that each chunk replica is to be striped across all the available nodes, with the restriction that no two replicas are in the same failure group.

  A write-affinity depth of 1 indicates that the first replica is written to the node writing the data. The second and third replicas are striped at the chunk level among the nodes that are farthest apart (based on failure group definition) from the node writing the data, with the restriction that no two replicas are in the same failure group. Using the rack topology example, the rack used for the first replica is not used by the second and the third replicas. This allows data to remain available even if the entire rack fails.

  A write-affinity depth of 2 indicates that the first replica is written to the node writing the data. The second replica is written to a node in the same rack as the first replica (based on the failure group definition), but in the other half of the rack. The third replica is striped at the chunk level across all available nodes that are farthest (per the failure group definition) from the nodes used by the first two replicas. Using the rack topology example, the second replica is placed in the same rack as the first replica, but in the second half of the rack. The third replica is striped across nodes that are not in the rack that is used by the first and second replicas.

  For compatibility reasons, the default write-affinity depth is 0 and the unit of striping is a block; however, if block group factor is specified, striping is done at the chunk level. The replica placement is done according to the specified policy at the best-effort level. If a

node does not have sufficient free space, IBM Spectrum Scale allocates space in other failure groups, ensuring that no two replicas of a chunk are assigned to the same node.

For MapReduce and data warehousing applications, a write-affinity depth of 1 is recommended, as the application benefits from the locality of the first replica of the block.

- **Write-affinity failure group**: The write-affinity failure group extends the write-affinity depth concept by allowing the application to control placement of each replica of a file and is therefore applicable in FPO-enabled environments only. It defines a policy that indicates the range of nodes where replicas of blocks of a particular file are to be written. This enables applications to control the layout of a file in the cluster to align with data access patterns. The write-affinity failure group specification function uses a failure group list to specify the nodes to be used for each of the replicas. A write-affinity failure group can be specified for each replica of a file.

  When the write-affinity failure group provides specification for all replicas, write-affinity depth is completely overridden. Write-affinity depth policy is used only for replica specification that is missing in the write-affinity failure group specification. The default policy is a null specification and uses the write-affinity depth settings for replica placement.

  The following format is used to specify write-affinity failure groups:

  ```
  FailureGroup1[;FailureGroup2[;FailureGroup3]
  ```

  where

  FailureGroupN
  identifies nodes on which to place the first, second, and third replicas of each chunk of data.
  FailureGroupN
  is a comma-separated string of up to three failure groups. Failure group list notation is extended to include more than one node or range of nodes in the following format:
  Rack1{:Rack2{:...{:Rackx}}},Location1{:Location2{:...{:Locationx}}},ExtLg1{:ExtLg2{:...{:ExtLgx}}}

  Wildcard characters (*) are supported in these fields.
  Range can be specified using '-'
  Specific numbers are specified using ':'
  If any part of the field is missing, it is interpreted as 0.

  For example, the attribute 1,1,1:2;2,1,1-3;2,0,* using the rack topology terminology indicates that the first replica is striped on rack 1, rack location 1, nodes 1 and 2; the second replica is striped on rack 2, rack location 1, nodes 1, 2 and 3; and the third replica is on rack 2, rack location 0, and all nodes in that location.

  The attribute 1,*,*; 2,*,*;3,*,* results in striping the first replica on all nodes in rack 1, the second replica on all nodes in rack 2, and the third replica on all nodes in rack 3.

  This attribute can be set using the policy rule or by setting the extended attribute write-affinity-failure-group using the **mmchattr** command on a file before writing data blocks.

- **File placement policy**: A file placement policy can be defined at the file system, storage pool, or file level of granularity. For this purpose, IBM Spectrum Scale has a policy engine that allows you to set data placement attributes based on the file name, fileset, and other attributes. The policy engine offers a rich set of language to query and search files, take action, and specify rules for data placement. In most cases, file placement policies apply to groups of files based on a set of attributes. Some of the placement attributes that can be affected by using policy rules include destination storage pool, chunk size, write-affinity depth, write-affinity failure group, and replication factor.

- **Recovery from failures**: In a typical FPO cluster, nodes have direct-attached disks. These disks are not shared between nodes as in a traditional Spectrum Scale cluster, so if the node is inaccessible, the associated disks are also inaccessible.

  IBM Spectrum Scale provides ways to automatically recover from these and similar common disk failure situations.

  In FPO environments, automated recovery from disk failures can be enabled using the *restripeOnDiskFailure=yes* configuration option with the **`mmchconfig`** command. Whether an FPO-enabled file system is a subject of an automated recovery attempt is determined by the max replication values for the file system. If either the metadata or data replicas are greater than one, a recovery action is triggered. The recovery actions are asynchronous, and IBM Spectrum Scale continues processing while the recovery takes place.

  The results from the recovery actions and any errors encountered are recorded in the Spectrum Scale logs. When auto-recovery is enabled, IBM Spectrum Scale delays recovery action for disk failures to avoid recovery from transient failures such as node reboot. The disk recovery wait period can be customized as needed. The default setting is 300 seconds for disks containing metadata, and 600 seconds for disks containing data only. The recovery actions include restoring proper replication of data blocks from the failed disks by creating additional copies on other available disks. In a temporary failure that occurs within the recovery wait period, recovery actions include the restarting of failed disks so they are available for allocation.

When using FPO file system, you can define policies for file placement. For a full list of policies, see the *GPFS V4.1: Advanced Administration Guide* SC23-7032-00 under the *Policy rules* section.

In this book, we created a basic FPO configuration. It is available in section 3.9, “Sample Spectrum Scale FPO configuration” on page 170.

3

Scenarios

This chapter provides information about multiple configuration scenarios using IBM Spectrum Scale and provides the necessary knowledge and know-how to implement the described scenarios, and to consider IBM Spectrum Scale as improvement to current setups that use other file systems or are clustered systems.

The following topics are covered in this chapter:

- IBM Spectrum Scale advantages over Network File System
- IBM Spectrum Scale in active-passive and mutual takeover clusters
- IBM Spectrum Scale in active-active clusters
- Two-node Linux IBM Spectrum Scale cluster
- Cluster NFS
- Windows operating system-only cluster
- Oracle Real Application Cluster with IBM Spectrum Scale
- IBM Spectrum Scale integration with IBM Spectrum Protect (formerly Tivoli Storage Manager)
- Sample Spectrum Scale FPO configuration
- Integrating with OpenStack Swift

© Copyright IBM Corp. 2015. All rights reserved.

3.1 IBM Spectrum Scale advantages over Network File System

IBM Spectrum Scale offers many advantages over Network File System (NFS), not only on the performance side, but also on resilience, scalability, security, and manageability. The following statements are applicable regardless of the NFS implementation:

- Spectrum Scale nodes connect to all the nodes in the Spectrum Scale cluster, all the time.
- On NFS clusters, the IP address needs to be moved around the cluster when either a problem occurs or doing service on the NFS server node. On Spectrum Scale no IP moves as the nodes connect to all the Spectrum Scale NSD servers, all the time.
- Metadata operations on NFS suffer from bottlenecks, as only the NFS server does the metadata operations. A simple listing files operation on an NFS directory with many files is very costly in time operations. NFS reads the inode, waits for the reply, and moves to the next inode, sequentially for all the files on the directory. Without showing any display until it has completed, the inode reads in all the directory from the stand-alone NFS server. All nodes in Spectrum Scale can do metadata operations locally.
- NFS trades between speed with no congestion control or congestion control less speed and higher latency (UDP or TCP). Tuning is possible (Nagle, MTU, aggregation, and so on) on NFS to mitigate part of those trades.
- Depending on the configuration. NFS locks the whole file for writing or does not lock anything. Spectrum Scale locks at block level for writing, regardless how big the file is, if only one block is going to be writing, only that block is locked for writing. Spectrum Scale only locks the entire file if the file is not in use.
- On some clusters that use the NFS as a resource, the NFS IP failover is often a pain point for the servers and the clients. Spectrum Scale does not do IP failovers.
- NFSv4 supports certain replication and parallel NFS access (pNFS).
- NFS does not provide any information lifecycle management (ILM). IBM Spectrum Scale has unparallel ILM capabilities.

For the above reasons among others, when NFS is used it is a good exercise to see how Spectrum Scale could help to improve the resilience, performance, and manageability of the setup. Very clear examples on which Spectrum Scale offers a clear benefit over NFS configurations, are not limited to the following:

- NAS
- SAP
- Multimedia libraries
- In general, any situation where NFS is used as a client or as a cluster

3.2 IBM Spectrum Scale in active-passive and mutual takeover clusters

The simplest active-passive cluster setup is a two-node scenario where one node runs the application while the other node is on cold standby. If there is a failover of the application, the storage, file systems, IP addresses, and applications are moved to the other node. During the failover, there is no service.

A mutual takeover cluster is active-passive cluster where at least two applications exist and the home node is different for each of them. So both nodes are doing something and also

both nodes are in cold standby for the other application. During the failover as in the active-passive cluster, there is no service for the application being moved.

IBM Spectrum Scale helps in this scenario to reduce the time to fail over to the standby node, which depends on the technology used, the number of disks, the type of storage, and many other factors. The time to fail over the storage and file system part of the cluster can take up to several minutes. When migrating to Spectrum Scale, the file system is already available in the standby node, so no operations are needed to achieve the availability of the failover. The IP move and application move are the only steps needed to move the service to the other node, not only speeding up the failover time, but also simplifying the cluster configuration.

Less failover time means less time when there is no service, and simpler cluster configurations mean fewer errors when managing those clusters.

3.3 IBM Spectrum Scale in active-active clusters

In some active-active clusters like DB2 PureScale, IBM Spectrum Scale comes embedded as a mandatory part of the software. In some others like Oracle Real Application Cluster (RAC), it is an optional component. Other solutions might come with their own approach on how to share the file systems.

Regardless of the technology, by using Spectrum Scale you add the unparallel information lifecycle management (ILM) capabilities, scalability, performance, maturity, and resilience that other solutions might or might not provide.

3.4 Two-node Linux IBM Spectrum Scale cluster

The following two series of steps describe how to install, configure, and start a two-node Linux on IBM POWER Spectrum Scale cluster. Use the steps in the following sections:

- Installing IBM Spectrum Scale
- Configure auxiliary tools
- Building the IBM Spectrum Scale portability layer on Linux nodes
- Create IBM Spectrum Scale cluster
- Create NSD disks
- Create the IBM Spectrum Scale file system

Note: This scenario intends to show a simple case of two Linux nodes Spectrum Scale cluster. It uses default settings for the cluster and file system creation. It is not intended to be the standard way to set up a two-node cluster.

3.4.1 Installing IBM Spectrum Scale

For this scenario, we use the following setup:

- Two logical partitions (LPARs) running on an IBM POWER7 system.
- The operative system (OS) is Red Hat Enterprise Linux 6.5.
- LPAR has access to five shared disks on the storage area network (SAN).
- All the requisites explained in Chapter 2, “Infrastructure planning and considerations” on page 29 are met.

Note: In our example, we installed the following packages on each node:

```
# yum install ksh perl gcc kernel-devel imake compat-libstdc++-33 gcc-c++
redhat-lsb ntp net-tools
```

In this scenario, we install IBM Spectrum Scale 4.1 TL1 Express Edition for Linux on IBM POWER. Both nodes are Spectrum Scale NSD servers. On each node, run the **rpm** command as shown in Example 3-1.

Example 3-1 Install IBM Spectrum Scale Express Edition for Linux on IBM POWER

```
# rpm -vih gpfs.base*.ppc64.rpm gpfs.docs*.noarch.rpm gpfs.gskit-*.ppc64.rpm gpfs.libsrc-*.noarch.rpm
gpfs.msg.en_US-*.noarch.rpm gpfs.src-*.noarch.rpm gpfs.gpl*.noarch.rpm
Preparing...                ########################################### [100%]
   1:gpfs.base              ########################################### [ 14%]
   2:gpfs.gpl               ########################################### [ 29%]
   3:gpfs.src               ########################################### [ 43%]
   4:gpfs.msg.en_US         ########################################### [ 57%]
   5:gpfs.libsrc            ########################################### [ 71%]
   6:gpfs.gskit             ########################################### [ 86%]
   7:gpfs.docs              ########################################### [100%]
```

Note: Always check that you have the latest service level of IBM Spectrum Scale when installing. If the IBM Spectrum Scale base level is not the latest, apply the latest available updates.

3.4.2 Configure auxiliary tools

In our setup, we configure only a remote shell and a Network Time Protocol (NTP) client. Other tools that help distributed environments can be used as well.

File /etc/hosts

In our example, we use the hosts file for name resolution. Both nodes have entries on their own `/etc/hosts` that cover the two nodes' names and IP addresses. Domain Name System (DNS) is also an option.

Remote shell

A remote shell and a way to copy files between the nodes is needed for Spectrum Scale. In our setup, we use Secure Shell (SSH) using the OpenSSH flavor.

Once OpenSSH is installed, it is needed to configure the passwordless access to all nodes in our setup. To do so, it is needed to generate the keys for each node and make all nodes aware of those keys so they can log in passwordless. One way to do so is shown in Example 3-2.

Note: On SSH server configuration files, the **PermitRootLogin** parameter must be set to *yes* to follow Example 3-2. Later on, it must be kept on *yes* or *without-password*.

Example 3-2 Exchanging OpenSSH keys

```
Generate the keys on both nodes:
[root@fslinux1 ~]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
```

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
be:5c:0d:3b:bb:9c:6b:ee:f8:3d:40:1b:ab:5e:67:f0 root@fslinux1
The key's randomart image is:
+--[ DSA 1024]----+
|                 |
|                 |
|                 |
|          o      |
|        S.o+     |
|       .  +*     |
|        ..=.E    |
|       ..*.B.    |
|       .==Xo..   |
+-----------------+
```

```
[root@fslinux2 ~]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
46:c9:35:09:7e:4e:da:e5:46:fd:85:e7:5e:34:06:47 root@fslinux2
The key's randomart image is:
+--[ DSA 1024]----+
|        ..o. ..E |
|       o o.. .o. |
|        = o o oo+|
|       . * +  .=o|
|        S o o   +|
|       .   .   ..|
|                .|
|                 |
|                 |
+-----------------+
```

Note: After the keys are generated, the public keys need to be distributed to both nodes. From node 1, we add our own public key authorized_keys file, copy it to node 2, add the node 2 public key, and copy it back to node 1.

```
[root@fslinux1 .ssh]# cat id_dsa.pub >> authorized_keys
[root@fslinux1 .ssh]# scp authorized_keys fslinux2:$PWD
The authenticity of host 'fslinux2 (172.16.20.158)' can't be established.
RSA key fingerprint is 5f:ff:31:73:00:44:f4:16:9d:e3:37:00:0c:fc:a4:db.
Are you sure you want to continue connecting (yes/no)? ^C[root@fslinux1 .ssh]# rm known_hosts
rm: remove regular file `known_hosts'? y
[root@fslinux1 .ssh]# scp authorized_keys fslinux2:$PWD
The authenticity of host 'fslinux2 (172.16.20.158)' can't be established.
RSA key fingerprint is 5f:ff:31:73:00:44:f4:16:9d:e3:37:00:0c:fc:a4:db.
Are you sure you want to continue connecting (yes/no)? ^C[root@fslinux1 .ssh]# rm known_hosts
rm: remove regular file `known_hosts'? y

[root@fslinux1 .ssh]# scp authorized_keys fslinux2:$PWD
The authenticity of host 'fslinux2 (172.16.20.158)' can't be established.
RSA key fingerprint is 5f:ff:31:73:00:44:f4:16:9d:e3:37:00:0c:fc:a4:db.
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'fslinux2,172.16.20.158' (RSA) to the list of known hosts.
root@fslinux2's password:
authorized_keys 100% 1222     1.2KB/s   00:00

[root@fslinux1 .ssh]# ssh fslinux2
Last login: Wed Oct 15 11:10:34 2014 from fslinux1
[root@fslinux2 ~]# cd .ssh
[root@fslinux2 .ssh]# cat id_dsa.pub >> authorized_keys
[root@fslinux2 .ssh]# scp authorized_keys fslinux1:$PWD
The authenticity of host 'fslinux1 (172.16.20.157)' can't be established.
RSA key fingerprint is 49:fd:ae:52:4b:53:62:38:65:27:a9:38:64:19:db:23.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'fslinux1,172.16.20.157' (RSA) to the list of known hosts.
root@fslinux1's password:
authorized_keys 100% 1825     1.8KB/s   00:00
```

Note: The banner in UNIX must be turned off. Create a special file in your home directory called `.hushlogin`. Otherwise, the Spectrum Scale commands show the banner.

NTP

Nodes need to have the same time. The easiest way to achieve that is to have them using the same NTP server. Linux includes an NTP client implementation. Regardless of which method you use, ensure that times are in synchronization. Example 3-3 shows how to check the times on the nodes.

Example 3-3 Checking time on nodes

```
[root@fslinux1 ~]# date ; ssh fslinux2 date
Wed Oct 15 11:39:42 EDT 2014
Wed Oct 15 11:39:42 EDT 2014
```

3.4.3 Building the IBM Spectrum Scale portability layer on Linux nodes

When Spectrum Scale runs on Linux nodes, the nodes need a portability layer to be able to run. The Spectrum Scale portability layer is a loadable kernel module that allows the Spectrum Scale daemon to interact with the operating system.

Note: The Spectrum Scale kernel module should be updated any time the Linux kernel is updated. Updating the Spectrum Scale kernel module after a Linux kernel update requires rebuilding and installing a new version of the module.

Use the following steps to build the Spectrum Scale portability layer on Linux nodes:

1. Check for the following before building the portability layer:
   - Updates to the portability layer at the *IBM Support Portal: Downloads for General Parallel File System* can be found at the following site:

     http://ibm.co/1BW4tmO
   - The latest kernel levels supported are in the IBM Spectrum Scale FAQ in the cluster IBM Knowledge Center:

     http://ibm.co/1G9j99v

– Or the IBM Spectrum Scale FAQ in the IBM Knowledge Center at:

  http://www-01.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html

2. Build your Spectrum Scale portability layer in one of three ways:
   – Using the **Autoconfig** tool (recommended).
   – Using the directions in /usr/lpp/mmfs/src/README.
   – If using IBM Spectrum Scale 4.1 TL1, a new tool called **mmbuildgpl** installs all the required prerequisites for the major Linux distributions using the default package manager (yum/zipper/dpkg).

Using the automatic configuration tool

The following example shows the commands required to build the Spectrum Scale portability layer using the automatic configuration option (**make Autoconfig**):

```
cd /usr/lpp/mmfs/src
make Autoconfig
make World
make InstallImages
```

Each kernel module is specific to a Linux version and platform. If you have multiple nodes running the same operating system level on the same platform, you can build the kernel module on one node, then create an RPM that contains the binary module for ease of distribution.

If you choose to generate an RPM package for portability layer binaries, perform the following additional step:

```
make rpm
```

When the command finishes, it displays the location of the generated RPM:

```
<...Last line of output...>
Wrote:
/usr/src/redhat/RPMS/x86_64/gpfs.gplbin-2.6.18-128.1.14.e15-3.3.0-1.x86_64.rpm
```

You can then copy the generated RPM package to other machines for deployment. The generated RPM can only be deployed to machines with identical architecture, distribution level, Linux kernel, and Spectrum Scale maintenance level.

Note: During the package generation, temporary files are written to the /tmp/rpm directory, so be sure there is sufficient space available. By default, the generated RPM goes to /usr/src/packages/RPMS/<arch> for SUSE Linux Enterprise Server and /usr/src/redhat/RPMS/<arch> for Red Hat Enterprise Linux.

Using the new mmbuildgpl tool

In our test environment, we have IBM Spectrum Scale 4.1 TL1 running so we can use the **mmbuildgpl** tool. Example 3-4 shows how to use this new tool to build the compatibility layer for Linux Spectrum Scale nodes.

Example 3-4 Compiling and installing Spectrum Scale GNU GPL portability layer

```
On each node:
# /usr/lpp/mmfs/bin/mmbuildgpl
------------------------------------------------------
mmbuildgpl: Building GPL module begins at Wed Oct 15 16:57:03 EDT 2014.
------------------------------------------------------
```

```
Verifying Kernel Header...
  kernel version = 2063299 (2.6.32-431.el6.ppc64, 2.6.32-431)
  module include dir = /lib/modules/2.6.32-431.el6.ppc64/build/include
  module build dir   = /lib/modules/2.6.32-431.el6.ppc64/build
  kernel source dir  = /usr/src/linux-2.6.32-431.el6.ppc64/include
  Found valid kernel header file under /lib/modules/2.6.32-431.el6.ppc64/build/include
Verifying Compiler...
  make is present at /usr/bin/make
  cpp is present at /usr/bin/cpp
  gcc is present at /usr/bin/gcc
  g++ is present at /usr/bin/g++
  ld is present at /usr/bin/ld
make World ...
make InstallImages ...
-------------------------------------------------------
mmbuildgpl: Building GPL module completed successfully at Wed Oct 15 16:57:19 EDT 2014.
-------------------------------------------------------
```

Note: Each time the Spectrum Scale code or the Linux kernel is updated, that layer needs to be rebuilt.

3.4.4 Create IBM Spectrum Scale cluster

To create our IBM Spectrum Scale cluster in our example, we create a file `linux.nodes` that has the content shown in Example 3-5.

Note: From now on in this example, unless explicitly stated, the Spectrum Scale commands are run in only one node of the cluster.

Example 3-5 linux.nodes file

```
fslinux1:quorum-manager
fslinux2:quorum-manager
```

To create a cluster, run the **`mmcrcluster`** command in only one node as shown in Example 3-6.

Note: Check that TCP ports 22 and 1191 are allowed both ways in all nodes. In this scenario, the Linux firewall was disabled.

Example 3-6 Creating IBM Spectrum Scale cluster

```
# mmcrcluster -N linux.nodes --ccr-enable -r /usr/bin/ssh -R /usr/bin/scp -C GPFS.ELASTIC -U STORAGE
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmcrcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note: In our example, the IBM Spectrum Scale binaries directory `/usr/lpp/mmfs/bin/` is exported to the **`PATH`** variable for root on both nodes.

After the cluster is created and when it is certain that the correct licenses are provided, change the license of the nodes accordingly. To do so, use the **mmchlicense** command. In this example, both nodes are NSD servers so run the **mmchlicense** command as shown in Example 3-7.

Example 3-7 Changing IBM Spectrum Scale license

```
# mmchlicense server -N fslinux1,fslinux2
The following nodes will be designated as possessing GPFS server licenses:
    fslinux1
    fslinux2
Please confirm that you accept the terms of the GPFS server Licensing Agreement.
The full text can be found at www.ibm.com/software/sla
Enter "yes" or "no": yes
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To check that the cluster is created, use the **mmlscluster** command as shown in Example 3-8.

Example 3-8 Listing the IBM Spectrum Scale cluster

```
# mmlscluster
GPFS cluster information
========================
  GPFS cluster name:         GPFS.ELASTIC
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name            IP address     Admin node name  Designation
--------------------------------------------------------------------------------
   1   fslinux1                   172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                   172.16.20.158  fslinux2         quorum-manager
```

3.4.5 Create NSD disks

In this setup, both nodes are NSD servers, and all storage is shared across both nodes. Before creating the NSD disks, check that both nodes have access to the shared disks. On Linux, one way to check the LUN serial numbers is shown in Example 3-9.

Example 3-9 Listing WWPN of shared disks

```
root@fslinux1 ~]# for MPATHDISK in `ls -1 /dev/dm-?`; do echo "$MPATHDISK SERIAL:"; /lib/udev/scsi_id
--page=0x83 --whitelisted --device=$MPATHDISK; done
/dev/dm-0 SERIAL:
/dev/dm-1 SERIAL:
/dev/dm-2 SERIAL:
3600507680191026c4000000000000157
/dev/dm-3 SERIAL:
3600507680191026c4000000000000158
/dev/dm-4 SERIAL:
3600507680191026c4000000000000159
/dev/dm-5 SERIAL:
3600507680191026c400000000000015b
/dev/dm-6 SERIAL:
3600507680191026c400000000000015a
```

```
/dev/dm-7 SERIAL:
3600507680191026c400000000000014f
/dev/dm-8 SERIAL:
/dev/dm-9 SERIAL:

[root@fslinux2 ~]# for MPATHDISK in `ls -1 /dev/dm-?`; do echo "$MPATHDISK SERIAL:"; /lib/udev/scsi_id
--page=0x83 --whitelisted --device=$MPATHDISK; done
/dev/dm-0 SERIAL:
/dev/dm-1 SERIAL:
/dev/dm-2 SERIAL:
3600507680191026c4000000000000157
/dev/dm-3 SERIAL:
3600507680191026c4000000000000159
/dev/dm-4 SERIAL:
3600507680191026c400000000000015a
/dev/dm-5 SERIAL:
3600507680191026c400000000000015b
/dev/dm-6 SERIAL:
3600507680191026c400000000000014f
/dev/dm-7 SERIAL:
3600507680191026c4000000000000158
/dev/dm-8 SERIAL:
/dev/dm-9 SERIAL:
```

This setup uses the multipath driver and has six shared disks across both nodes. To create the NSD disks, a file with the NSD definition must be created. In this example, only three disks are used, all in the same failure group and in the *system* pool. We use the stanza file called `disks.nsd` and its content is shown in Example 3-10.

Example 3-10 disks.nsd file

```
%nsd:
 device=/dev/dm-2
 nsd=NSD001
 servers=fslinux1,fslinux2
 usage=dataAndMetadata
 failureGroup=1
 pool=system

%nsd:
 device=/dev/dm-3
 nsd=NSD002
 servers=fslinux2,fslinux1
 usage=dataAndMetadata
 failureGroup=1
 pool=system

%nsd:
 device=/dev/dm-4
 nsd=NSD003
 servers=fslinux1,fslinux2
 usage=dataAndMetadata
 failureGroup=1
 pool=system
```

Note: Please notice that the order of servers on NSD is different for NSD002. The idea is to spread the load of the NSD across all the servers. Ideally you would like to spread the NSD alternating the primary server on each. The more NSD and servers the more spread the load.

To create the NSD disks, the **mmcrnsd** command must be used. The input file from Example 3-10 is used. See Example 3-11 on page 107 which shows how the **mmcrnsd** command is used to create the NSD disks, and the **mmlsnsd** command to list the NSD disks.

Example 3-11 Creating and listing NSD

```
# mmcrnsd -F disks.nsd
mmcrnsd: Processing disk dm-2
mmcrnsd: Processing disk dm-3
mmcrnsd: Processing disk dm-4
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

Listing the NSD disks:
# mmlsnsd -L -a

 File system   Disk name    NSD volume ID      NSD servers
------------------------------------------------------------------------------------------
 (free disk)   NSD001       149DAC10543EF1ED   fslinux1,fslinux2
 (free disk)   NSD002       149DAC10543EF1F0   fslinux2,fslinux1
 (free disk)   NSD003       149DAC10543EF1F3   fslinux1,fslinux2
```

3.4.6 Starting the IBM Spectrum Scale cluster

At this moment, we have a two-node cluster setup with three NSD disks defined. Before creating the Spectrum Scale file system, start the cluster with the **mmstartup** command and list the status with the **mmgetstate** command. Example 3-12 shows how to start the cluster in all nodes and get the state of the nodes.

Example 3-12 Starting and listing IBM Spectrum Scale cluster status

```
# mmstartup -a
Thu Oct 16 09:37:50 EDT 2014: mmstartup: Starting GPFS ...
# mmgetstate -L -a

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      fslinux1           2        2          2       active      quorum node
       2      fslinux2           2        2          2       active      quorum node
```

Note: In case you run into problems, check the Spectrum Scale logs, which are in the `/var/adm/ras` directory. For detailed problem determination information, see Chapter 8, “Problem determination” on page 411.

3.4.7 Quorum configuration

The setup now has two nodes and a quorum of two, which means that if any of the two nodes are not active, quorum is not met. If you are forced to use only two nodes, and need to have the file system up even with one node down, you have to use the tiebreaker disk to uneven the quorum using the **mmchconfig**, **mmlsconfig**, and **mmgetstate** commands as shown in Example 3-13.

Example 3-13 Adding tiebreaker disks and listing quorum status

```
# mmgetstate -L -a -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
```

```
---------------------------------------------------------------------------------
       1      fslinux1           2        2          2       active      quorum node
       2      fslinux2           2        2          2       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:           2
Number of local nodes active in the cluster:      2
Number of remote nodes joined in this cluster:    0
Number of quorum nodes defined in the cluster:    2
Number of quorum nodes active in the cluster:     2
Quorum = 2, Quorum achieved
```

Note: Quorum is 2 and we have both nodes up, so quorum is met.

```
# mmchconfig tiebreakerDisks="NSD001;NSD002;NSD003"
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note: In this setup, all three NSD disks were added as tiebreakers.

```
# mmgetstate -L -a -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
---------------------------------------------------------------------------------
       1      fslinux1           1        2          2       active      quorum node
       2      fslinux2           1        2          2       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:           2
Number of local nodes active in the cluster:      2
Number of remote nodes joined in this cluster:    0
Number of quorum nodes defined in the cluster:    2
Number of quorum nodes active in the cluster:     2
Quorum = 1*, Quorum achieved
```

Note: Now quorum is 1 and we have achieved 2 nodes up and quorum. On the cluster configuration, see the NSD disks that are tiebreakers.

```
# mmlsconfig
Configuration data for cluster GPFS.ELASTIC:
--------------------------------------------
clusterName GPFS.ELASTIC
clusterId 2497146759570973977
autoload no
uidDomain STORAGE
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks NSD001;NSD002;NSD003
adminMode central

File systems in cluster GPFS.ELASTIC:
-------------------------------------
```

```
(none)
```

Note: Adding and removing tiebreaker disks can be done online when the cluster is in cluster configuration repository (CCR) mode.

3.4.8 Start IBM Spectrum Scale at boot

In this setup, we want the IBM Spectrum Scale cluster to come up at boot time. In other scenarios, that might not be the wanted setup. To enable IBM Spectrum Scale to start at boot, use the **mmchconfig** command. To list the current value of the autoload attribute, use the **mmlsconfig** command. See Example 3-14.

Example 3-14 Setting autoload to yes

```
# mmchconfig autoload=yes
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
# mmlsconfig
Configuration data for cluster GPFS.ELASTIC:
--------------------------------------------
clusterName GPFS.ELASTIC
clusterId 2497146759570973977
uidDomain STORAGE
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks NSD001;NSD002;NSD003
autoload yes
adminMode central

File systems in cluster GPFS.ELASTIC:
-------------------------------------
(none)
```

3.4.9 Create the IBM Spectrum Scale file system

In this example, only the simplest file system is created by using the defaults for all parameters and the input file from Example 3-10 on page 106. To create a Spectrum Scale file system, the **mmcrfs** command must be used. Example 3-15 shows how the cluster is started in all nodes with the **mmstartup** command, and then how the Spectrum Scale file system is created and listed.

Example 3-15 Creating a Spectrum Scale file system

```
# mmcrfs myFS -F nsd.file -T /shared
The following disks of myFS will be formatted on node fslinux1:
    NSD001: size 10240 MB
    NSD002: size 10240 MB
    NSD003: size 10240 MB
Formatting file system ...
Disks up to size 101 GB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
```

```
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/myFS.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To mount the file system use the **mmmount** command. To check whether the file system is mounted, use either the operating system commands or the **mmlsmount** command. See Example 3-16.

Example 3-16 Mounting and checking mount on the Spectrum Scale file system

```
# mmmount myFS -a
Thu Oct 16 10:29:43 EDT 2014: mmmount: Mounting file systems ...
# mmlsmount myFS -L
File system myFS is mounted on 2 nodes:
  172.16.20.157   fslinux1
  172.16.20.158   fslinux2
```

Now you can use the file system normally as any other POSIX fully compliant file system.

3.5 Cluster NFS

IBM Spectrum Scale has cluster NFS (cNFS) built-in capabilities that allow building an NFS cluster with IP failover between nodes without the need of any other software but IBM Spectrum Scale. Although, in most cases native Spectrum Scale access via NSD protocol has multiple advantages over NFS as explained in section 3.1, “IBM Spectrum Scale advantages over Network File System” on page 98, in some cases NFS access to a Spectrum Scale file system might be needed.

In this scenario, we use two Linux LPARs running on Power Systems as NSD servers that get their storage from SAN. Each node has an NFS IP address, and four Intel nodes use the storage provided by the cNFS to store virtual machines.

3.5.1 NFS setup

Depending on your base installation, NFS utils has to be installed. To do so, run on each NSD server that will be serving cNFS the **yum** command as shown in Example 3-17.

Example 3-17 Install NFS

```
#yum install nfs-utils
```

Once installed, as Spectrum Scale is going to manage the NFS daemons, the daemons do not need to be started by the operating system at boot. Although, other services need to be started so Spectrum Scale cNFS can function. Example 3-18 shows the **chkconfing** commands that need to run on each node that will act as the cNFS server.

Example 3-18 Configuring services for cNFS on each cNFS server

```
# chkconfig nfs off
# chkconfig nfslock off
# chkconfig rpcbind on
# chkconfig --level 345 rpcgssd on
# chkconfig rpcidmapd on
```

```
# chkconfig rpcsvcgssd on
```

3.5.2 Configuring cNFS

First, configure the control directory. This directory is used by all nodes that are cNFS servers to control HA and the status on the NFS exported directories. To do so, use the **mmchconfig** command as shown Example 3-19 from any node with administrative privileges in the Spectrum Scale cluster.

Note: To run **mmchconfig cnfsSharedRoot=<directory>**, Spectrum Scale has to be down in all nodes.

Example 3-19 mmchconfig defining cNFS directory

```
# mmchconfig cnfsSharedRoot=/shared/cNFSroot
Verifying GPFS is stopped on all nodes ...
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note: Use preferably on a small separate Spectrum Scale file system that is not exported by NFS. The Spectrum Scale file system that contains the directory must be configured to be mounted automatically upon the start of Spectrum Scale on all the cNFS nodes (**-A** yes option on the **mmchfs** command). **cnfsSharedRoot** is a mandatory parameter and must be defined first.

After configuring cnfsSharedRoot, it is possible to start Spectrum Scale normally.

To configure the IP address, the **mmchnode** command is used with the **--cnfs-interface** parameter. It is a per node attribute and we can have more than one cNFS IP address per node. Example 3-20 shows how to set up the cNFS IP addresses per node.

Note: This approach uses IP alias to set up the cNFS service IP addresses.

Example 3-20 Configuring cNFS IP addresses

```
# mmchnode --cnfs-interface="10.10.12.19" -N GPFSnsd01
Wed Dec 17 09:59:06 EET 2014: mmchnode: Processing node GPFSnsd01
Restarting monitor
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmchnode --cnfs-interface="10.10.12.20" -N GPFSnsd02
Wed Dec 17 10:00:07 EET 2014: mmchnode: Processing node GPFSnsd02
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

At this point, the alias IP addresses should be up on the nodes. To see the cNFS IP addresses and other configurations, use the **mmlscluster** command with the **--cnfs** parameter as shown in Example 3-21.

Example 3-21 Listing cNFS with mmlscluster command

```
# mmlscluster --cnfs

GPFS cluster information
========================
  GPFS cluster name: Elastic.Storage
  GPFS cluster id:           3284823492711811309

Cluster NFS global parameters
-----------------------------
  Shared root directory:                /shared/cNFSroot
  rpc.mountd port number:               (undefined)
  nfsd threads:                         32
  Reboot on failure enabled:            yes
  CNFS monitor enabled:                 yes

 Node  Daemon node name            IP address       CNFS state  group  CNFS IP address list
-------------------------------------------------------------------------------------------
   1   GPFSnsd01                   10.10.12.92      enabled        0   10.10.12.19
   2   GPFSnsd02                   10.10.12.93      enabled        0   10.10.12.20
```

Note: Depending on the type of load, you might need to change the *nfsd threads*. Also, if cNFS is not a critical workload compared with other Spectrum Scale loads in the NSD server, which is also the cNFS server, evaluate and change *Reboot on failure enabled* to *no*.

From Spectrum Scale cNFS perspective, all is configured now. However, manually duplicate the `/etc/exports` consistent file across all the cNFS nodes.

3.6 Windows operating system-only cluster

There are several steps to installing IBM Spectrum Scale on Windows operating system nodes in addition to fulfilling the software and hardware detailed on Chapter 2, “Infrastructure planning and considerations” on page 29. For Windows operating system nodes on Windows operating system-only clusters or not, prepare the installation of IBM Spectrum Scale with the next steps:

- Configuring Windows operating systems
  - Static IP address 3.6.2, “Static IP address” on page 113.
  - Active Directory domain 3.6.3, “Active Directory domain” on page 114.
  - Account Control (UAC) 3.6.4, “UAC” on page 114.
  - Disable IPv6 3.6.5, “Disable IPv6” on page 115.
  - Windows firewall 3.6.6, “Windows operating system firewall” on page 118.
  - IBM Spectrum Scale traces auxiliary tools 3.6.7, “IBM Spectrum Scale traces auxiliary tools” on page 120.
- Installing Cygwin

This scenario has two Windows operating system nodes running Windows Server 2012 R2. In this setup, both nodes are also AD domain servers but it is not a requisite that the AD domain servers are part of the IBM Spectrum Scale cluster. See Figure 3-1 on page 113.

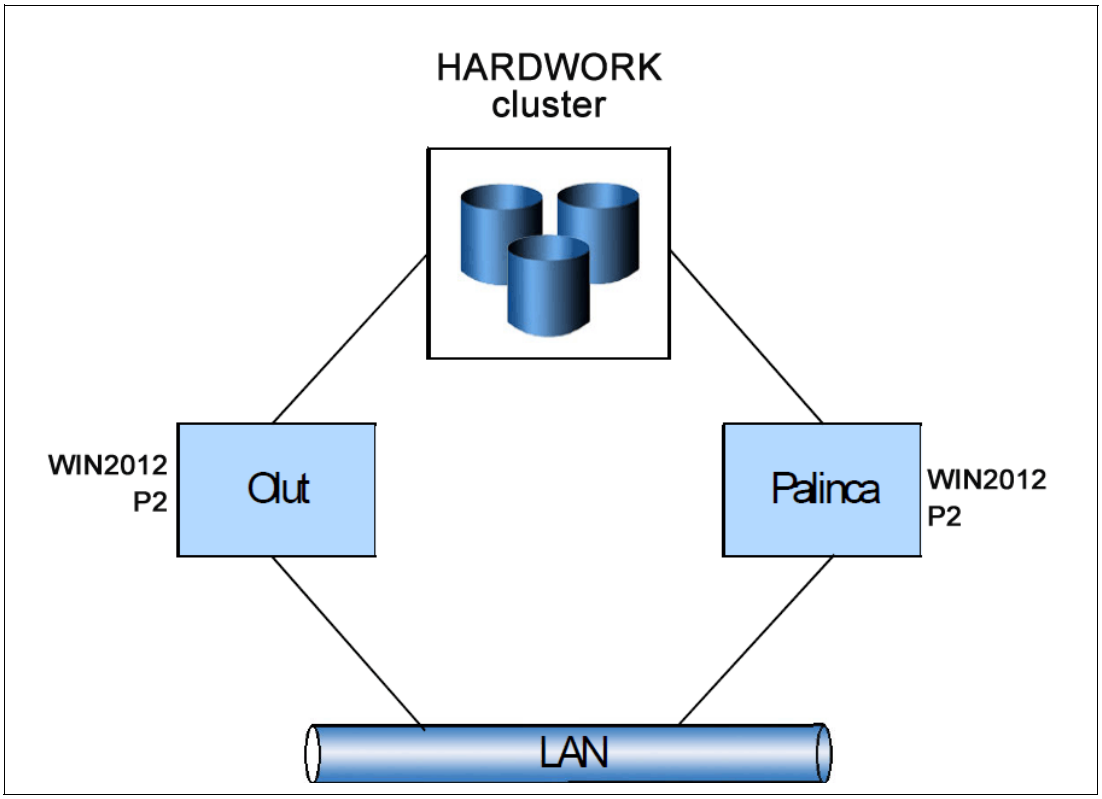


Figure 3-1 Windows operating system-only cluster

Note: See the IBM Spectrum Scale FAQ in the cluster IBM Knowledge Center:

http://ibm.co/1G9j99v

Or, see:

IBM Spectrum Scale FAQ in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html

List the levels of the Windows operating system supported by IBM Spectrum Scale.

3.6.1 Configuring Windows operating system

This topic provides details about installing and configuring Windows operating systems on systems that are added to an IBM Spectrum Scale cluster.

3.6.2 Static IP address

IBM Spectrum Scale communication requires invariant static IP addresses for each Spectrum Scale node.

3.6.3 Active Directory domain

All Windows systems in the same Spectrum Scale cluster should be members of the same Active Directory (AD) domain. The nodes must join to the Windows operating system domain before adding them to a Spectrum Scale cluster.

If you need to configure an AD server or cluster, check the Microsoft online documentation for further support:

http://bit.ly/1IDpgQe

3.6.4 UAC

On Windows Server 2008 nodes, you must disable UAC for Spectrum Scale to operate correctly. UAC needs to be disabled for the entire system, not just turned off for Spectrum Scale administrative users.

Windows Server 2008 R2 and Windows Server 2012 (including R2) do not have this requirement. However, if UAC is enabled, some Spectrum Scale functions might not work properly. Therefore, it is recommended that UAC be disabled, regardless of the Windows operating system version.

To disable UAC on a Windows Server 2012 R2 systems, follow these steps:

1. Open the *System Configuration* application under *Administrative Tools*.
2. Select the *Tools* tab, scroll down to select *Change UAC Settings,* and click **Launch**. See Figure 3-2.
3. Select the UAC vertical bar, lower it to the *Never notify* setting, click **OK**. *See* Figure 3-3 on page 115.
4. Select **OK** back on the *Tools tab*.
5. Reboot the server.

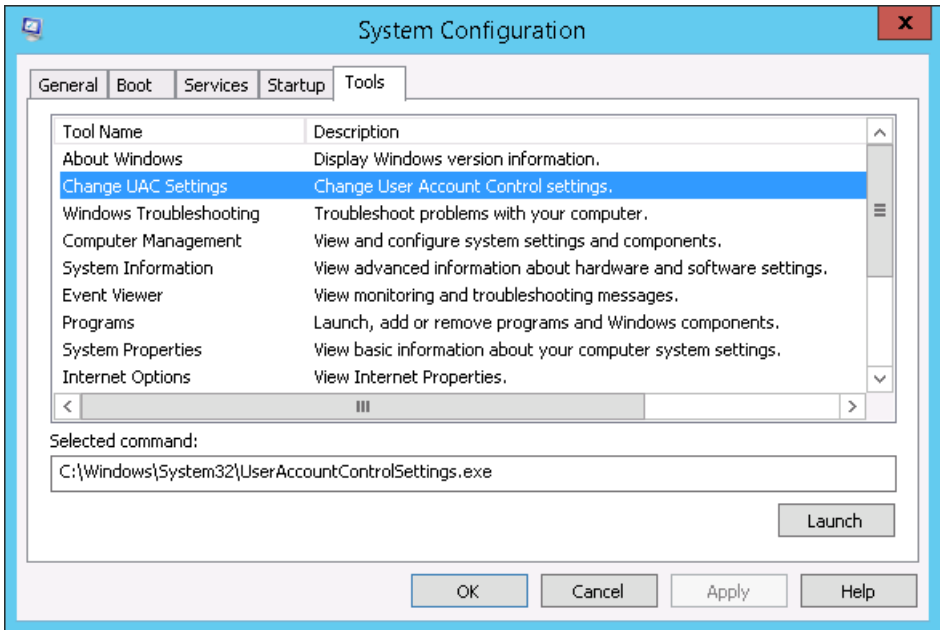


Figure 3-2 Tools tab

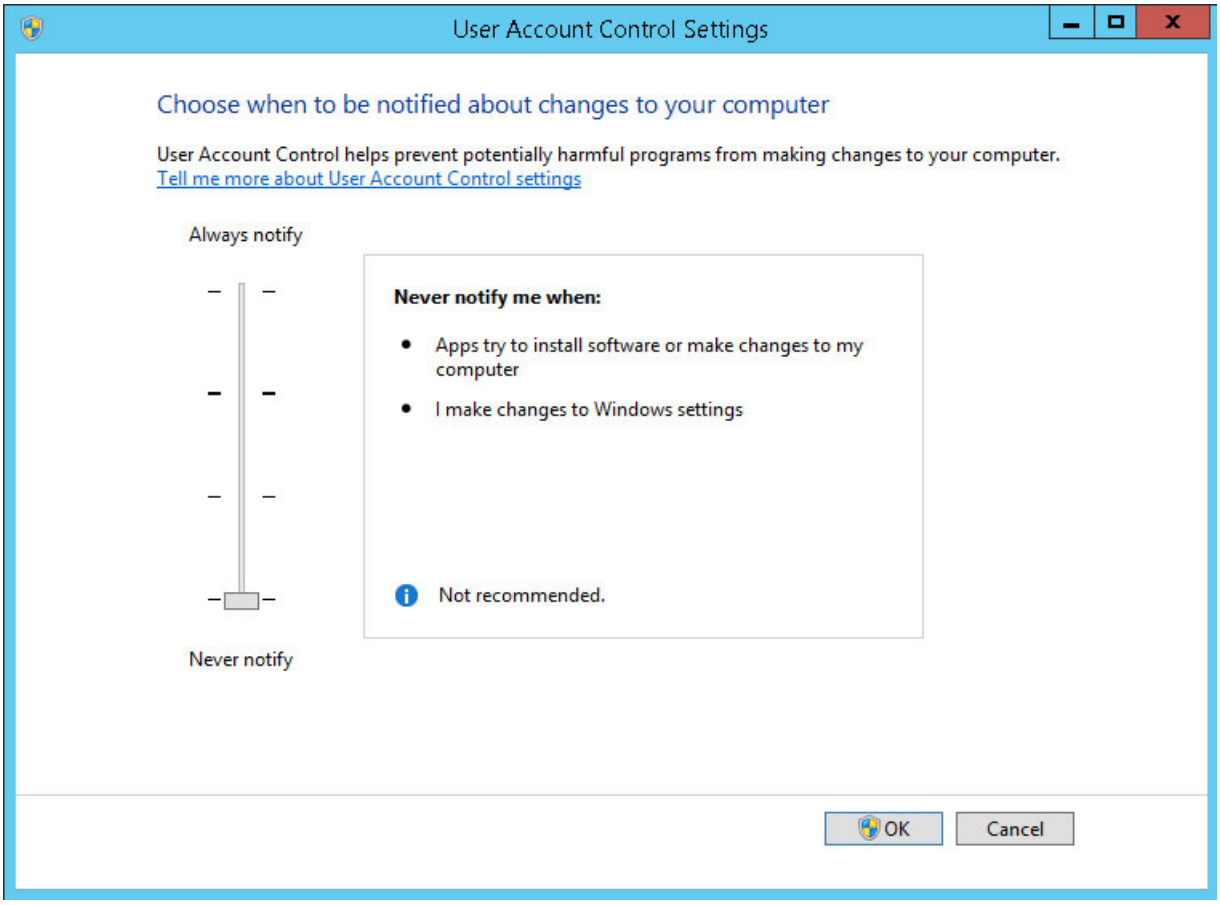

User Account Control Settings
Choose when to be notified about changes to your computer
User Account Control helps prevent potentially harmful programs from making changes to your computer.
Tell me more about User Account Control settings
Always notify
Never notify me when:
Apps try to install software or make changes to my computer
I make changes to Windows settings
Not recommended.
Never notify
OK
Cancel

Figure 3-3 UAC settings window

3.6.5 Disable IPv6

In addition to disabling IPv6 on the *Device Properties* as shown on Figure 3-4 on page 116, IPv6 should be disabled on the Windows operating system registry as well.

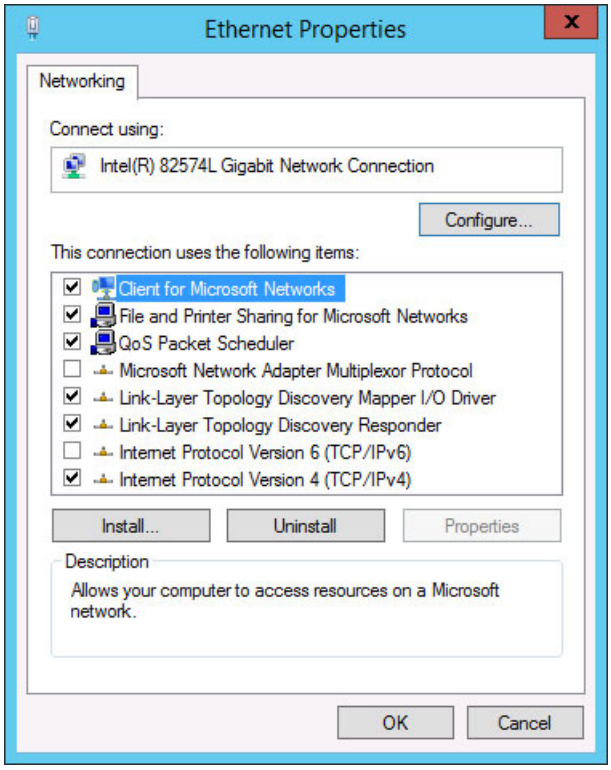


Figure 3-4 Device properties window

To disable IPv6 in the Windows operating system registry, open regedit.exe as *Administrator* or user with enough rights to do so. On `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP6\Parameters`, create a DWORD key named *DisabledComponents* as shown on Figure 3-5 and Figure 3-6 on page 117.

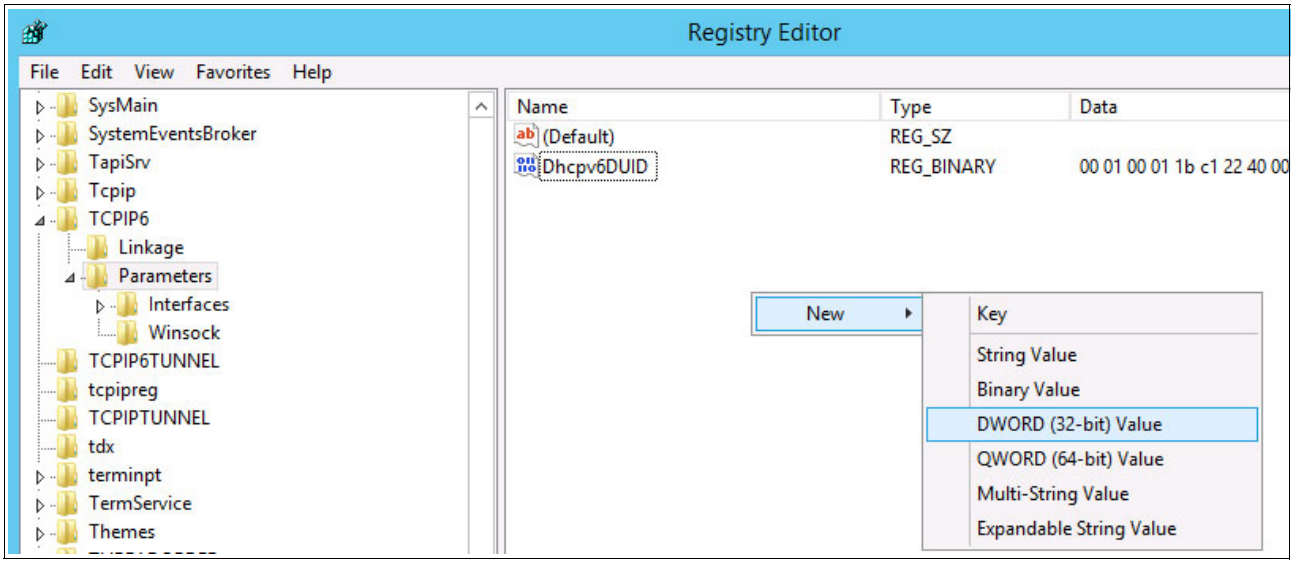


Figure 3-5 Creating a DWORD

Figure 3-6 shows how to name the property.

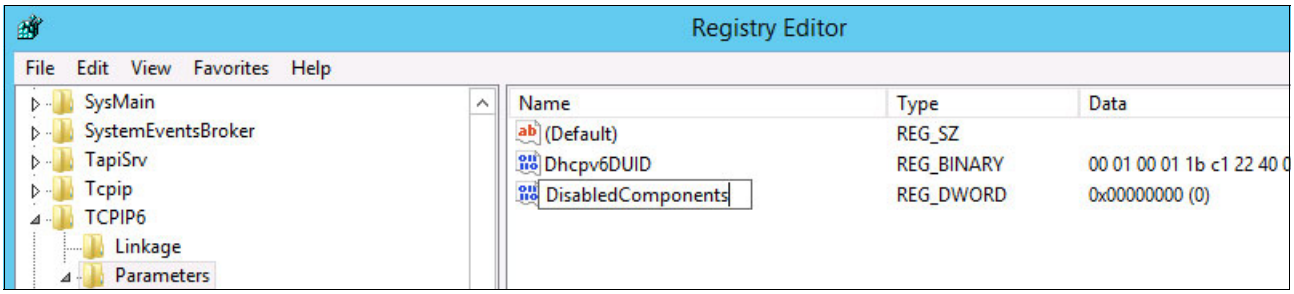


Figure 3-6 Naming the DWORD to DisabledComponents

Once created, edit the value by right-clicking **DisabledComponents** and select the **Modify Binary Data** option as shown in Figure 3-7.

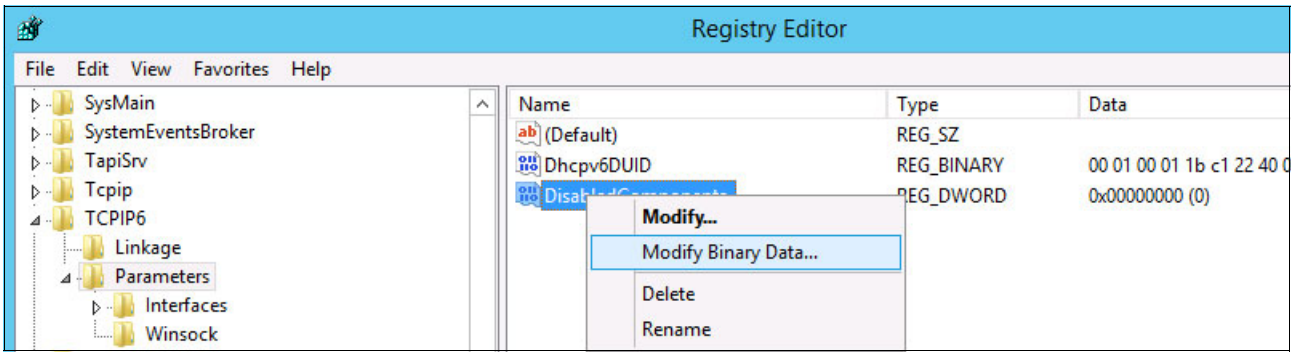


Figure 3-7 Editing the DWORD binary value

On the new window that is displayed, insert *FF FF FF FF* as the value and press **OK**. See Figure 3-8.

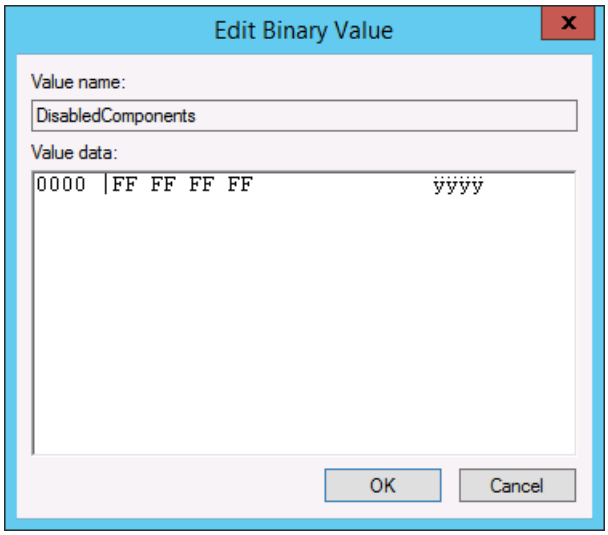


Figure 3-8 Entering the DWORD value

Now the key has a value of 0xffffffff in the Windows operating system registry editor as shown in Figure 3-9 on page 118.

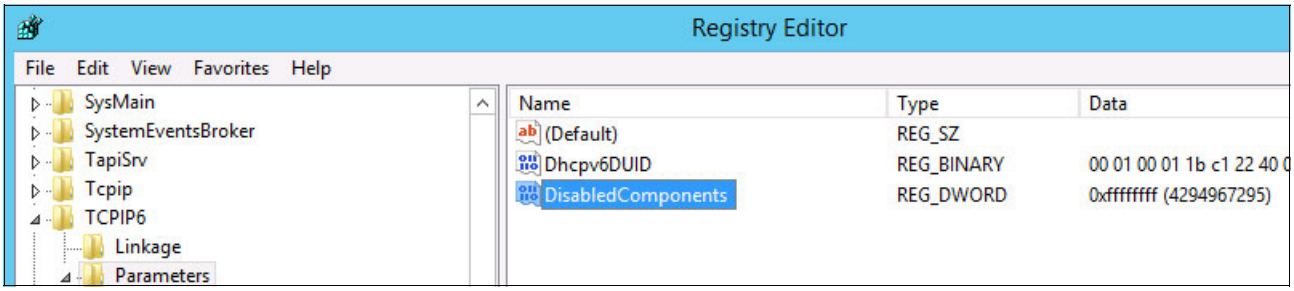


Figure 3-9 Showing DisabledComponents value

Reboot the Windows operating system server.

3.6.6 Windows operating system firewall

Spectrum Scale requires that you modify the default Windows operating system firewall settings to allow TCP 1191 (and TCP 22, if a heterogeneous cluster) port. The simplest change that allows Spectrum Scale to operate properly is to disable the firewall.

Open the Windows operating system firewall in Control Panel (see Figure 3-10 on page 119) and click **Turn Windows Firewall on or off**.

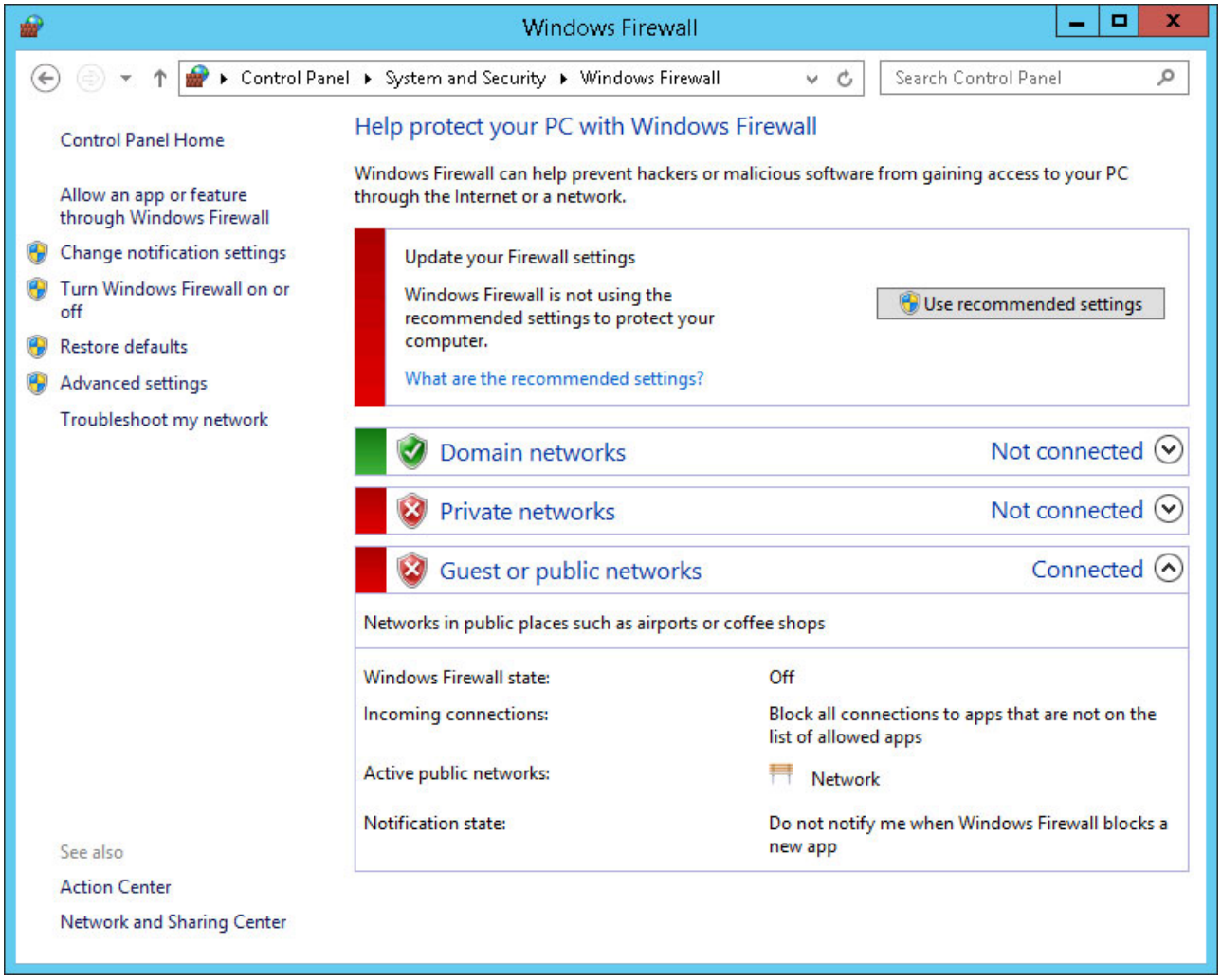

Windows Firewall
Control Panel ▸ System and Security ▸ Windows Firewall
Search Control Panel
Control Panel Home
Allow an app or feature through Windows Firewall
Change notification settings
Turn Windows Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network
Help protect your PC with Windows Firewall
Windows Firewall can help prevent hackers or malicious software from gaining access to your PC through the Internet or a network.
Update your Firewall settings
Windows Firewall is not using the recommended settings to protect your computer.
Use recommended settings
What are the recommended settings?
Domain networks
Not connected
Private networks
Not connected
Guest or public networks
Connected
Networks in public places such as airports or coffee shops
Windows Firewall state:
Off
Incoming connections:
Block all connections to apps that are not on the list of allowed apps
Active public networks:
Network
Notification state:
Do not notify me when Windows Firewall blocks a new app
See also
Action Center
Network and Sharing Center

Figure 3-10 Control Center Firewall main panel

Select **Off** for at least the domain networks. In this scenario, the firewall is completely disabled as shown in Figure 3-11 on page 120. Click **OK**.

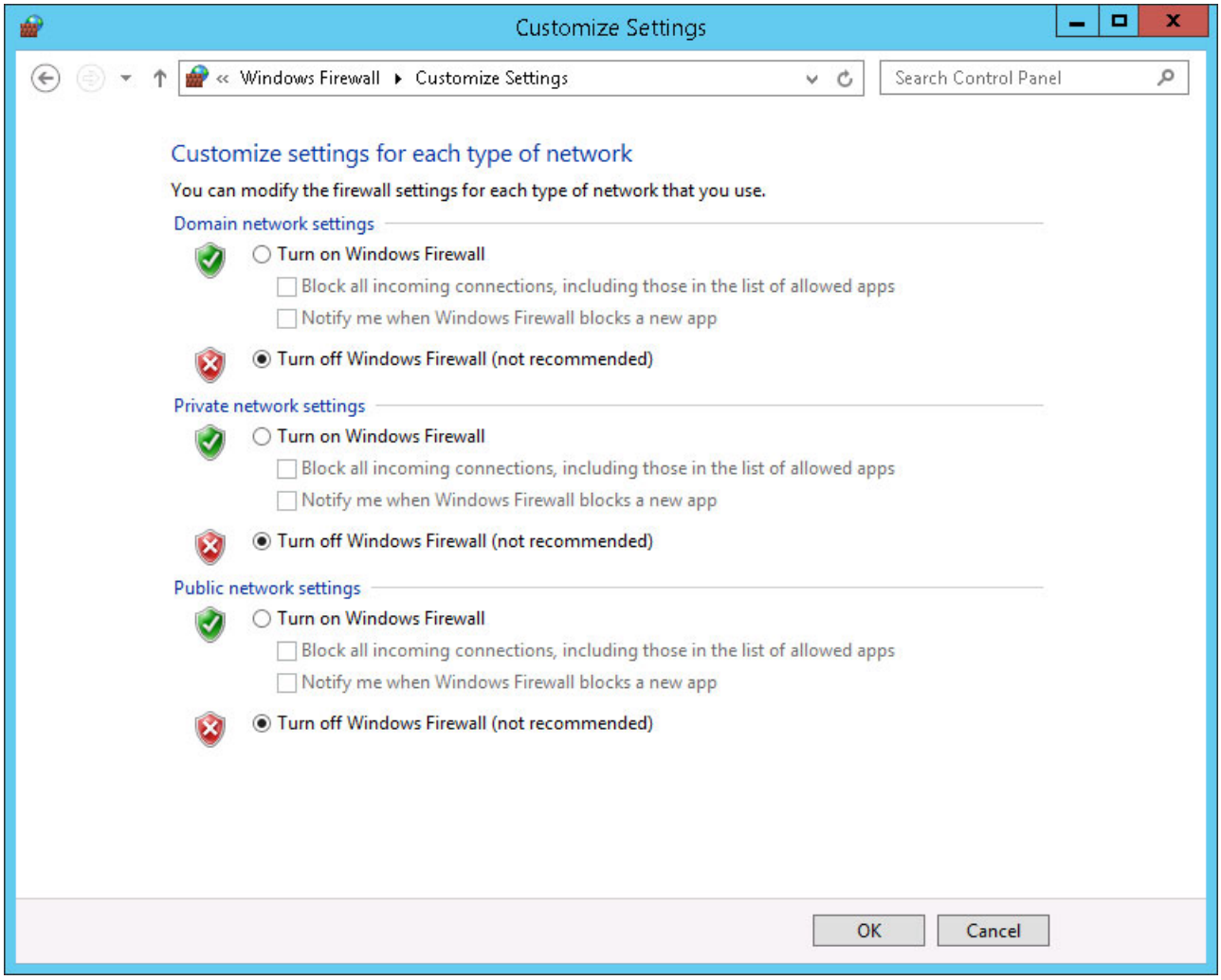

Customize Settings
Windows Firewall ▸ Customize Settings
Search Control Panel
Customize settings for each type of network
You can modify the firewall settings for each type of network that you use.
Domain network settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed apps
Notify me when Windows Firewall blocks a new app
Turn off Windows Firewall (not recommended)
Private network settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed apps
Notify me when Windows Firewall blocks a new app
Turn off Windows Firewall (not recommended)
Public network settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed apps
Notify me when Windows Firewall blocks a new app
Turn off Windows Firewall (not recommended)
OK
Cancel

Figure 3-11 Disabled firewall for all networks

3.6.7 IBM Spectrum Scale traces auxiliary tools

IBM Spectrum Scale diagnostic tracing (**mmtracectl**) on Windows operating system uses the Microsoft programs called **tracefmt.exe** and **tracelog.exe**. These programs are not included with Windows operating system but can be downloaded from Microsoft. You only need tracefmt.exe and tracelog.exe for tracing support; it is not required for normal Spectrum Scale operations.

To allow IBM Spectrum Scale diagnostic tracing on Windows operating system using the Windows Driver Kit (WDK), follow these steps:

1. Download the WDK from Microsoft. Information about how to do a stand-alone tool set installation can be found at Microsoft Developer Network (MSDN):

   http://bit.ly/XRdQrL

2. Install the Tools feature of the WDK on some system to obtain a copy of tracefmt.exe and tracelog.exe. The programs are in the `tools\tracing\amd64` directory.
3. Copy tracefmt.exe and tracelog.exe to the `%SystemRoot%` directory (for example, `C:\Windows`) or some other directory included in the **PATH** environment variable for all users.

3.6.8 Installing Cygwin

Cygwin must be installed before installing IBM Spectrum Scale. It is a software package that provides a UNIX like environment on Windows operating systems and provides runtime support for POSIX applications and includes programs such as **grep**, **ksh**, **ls**, and **ps**. Cygwin can be obtained from the Cygwin website:

https://cygwin.com/install.html

Note: To avoid problems with NTFS ACL, always join to the AD domain before installing Cygwin. This is particularly important when Cygwin is being installed in AD servers into the C: drive.

Install Cygwin 32-bits version. When running Cygwin setup, only the standard packages are installed by default. IBM Spectrum Scale requires installation of additional packages, which are listed in the installation steps that follows:

- flip: Convert text file line endings between UNIX and DOS formats
- m4: GNU implementation of the traditional UNIX macro processor
- mksh: MirBSD Korn Shell
- perl: Larry Wall's Practical Extracting and Report Language
- procps: System and process monitoring utilities
- openssh: The OpenSSH server and client programs (only required for having UNIX or Linux nodes in the cluster)

Note: Throughout this information, UNIX file name conventions are used. For example, the IBM Spectrum Scale cluster configuration data is stored in the `/var/mmfs/gen/mmsdrfs` file. On Windows operating systems, the UNIX name space starts under the Cygwin installation directory, which by default is *%SystemDrive%\cygwin*, so the IBM Spectrum Scale cluster configuration data is stored in the `C:\cygwin\var\mmfs\gen\mmsdrfs` file.

3.6.9 Install IBM Spectrum Scale

IBM provides Spectrum Scale as Windows operating system installer packages (MSI), which allow both interactive and unattended installations. Perform the IBM Spectrum Scale installation steps as the administrator or some other member of the administrators group.

Note: When installing IBM Spectrum Scale, always check that you have the latest service levels. If the IBM Spectrum Scale base level is not the latest, apply the latest available updates.

In this scenario, we use interactive installation on both nodes. The IBM Spectrum Scale version that it is installed is the Standard Edition.

To install IBM Spectrum Scale, on each node select the appropriate license, in this case the `gpfs.ext-4.1-Windows.license.msi` file as shown in Figure 3-12 on page 122.

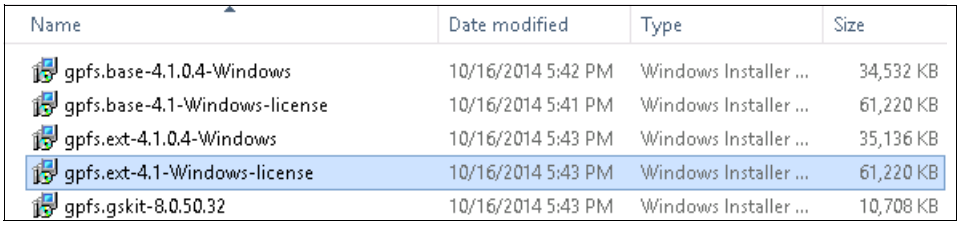


Figure 3-12 IBM Spectrum Scale Windows operating system license file

1. Select **Next** on the welcome panel of the IBM Spectrum Scale license installation program, as shown in Figure 3-13.

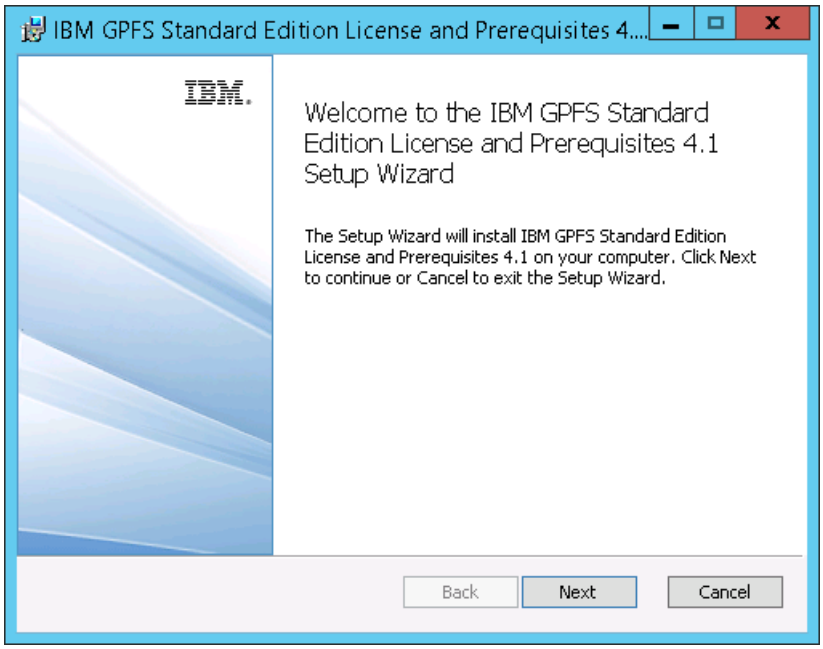

IBM GPFS Standard Edition License and Prerequisites 4....
IBM.
Welcome to the IBM GPFS Standard Edition License and Prerequisites 4.1 Setup Wizard
The Setup Wizard will install IBM GPFS Standard Edition License and Prerequisites 4.1 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

Figure 3-13 Welcome panel of IBM Spectrum Scale Standard Edition license

2. When all the checks are passed, the installation panel is displayed. Then, click **Install**. See Figure 3-14 on page 123.

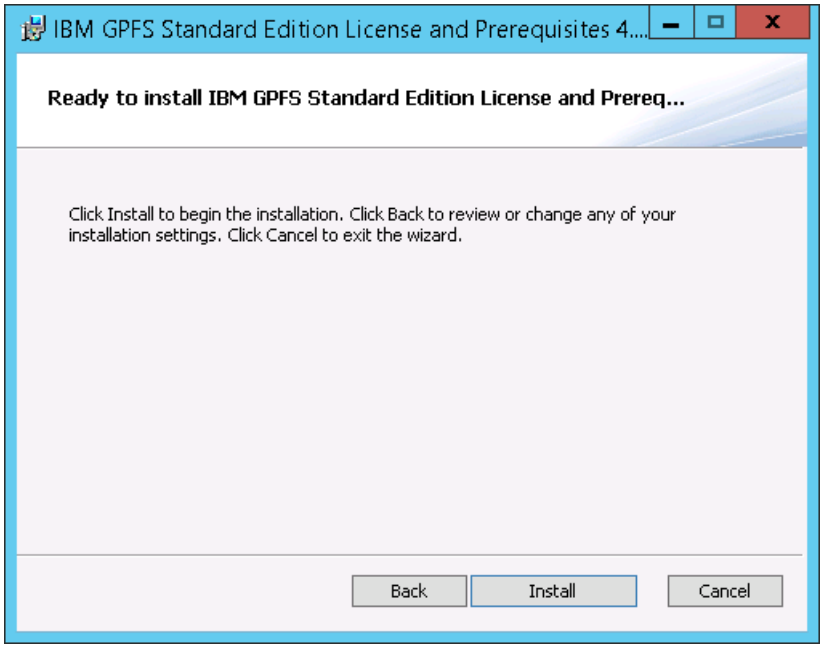

IBM GPFS Standard Edition License and Prerequisites 4....
Ready to install IBM GPFS Standard Edition License and Prereq...
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

Figure 3-14 Installation panel

3. During the installation, the license panel is displayed. Click **Accept** to accept the license as shown in Figure 3-15.

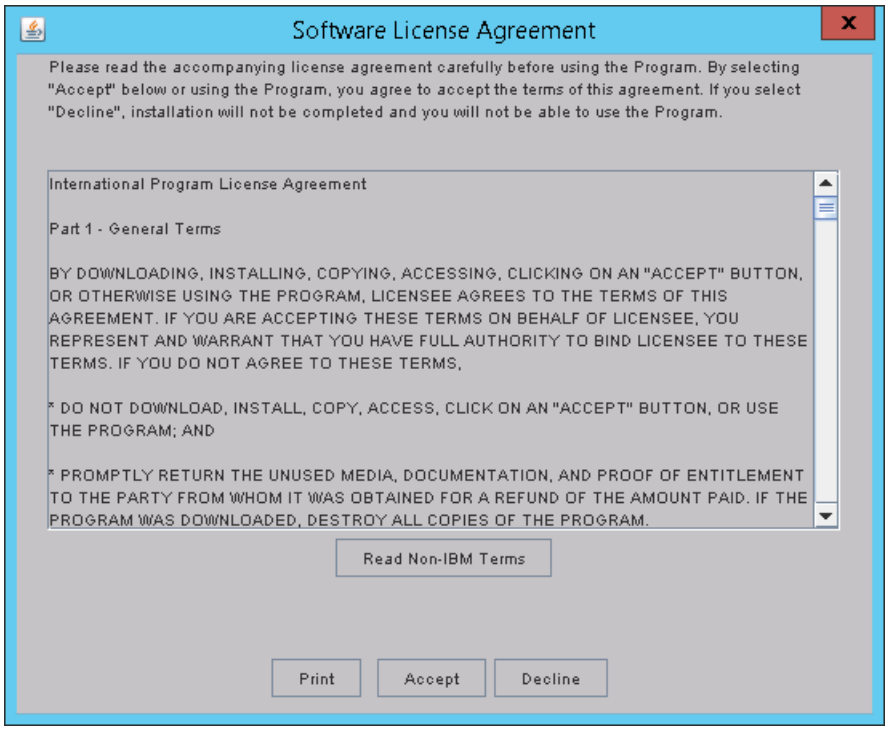

Software License Agreement
Please read the accompanying license agreement carefully before using the Program. By selecting "Accept" below or using the Program, you agree to accept the terms of this agreement. If you select "Decline", installation will not be completed and you will not be able to use the Program.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, CLICKING ON AN "ACCEPT" BUTTON, OR OTHERWISE USING THE PROGRAM, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF LICENSEE, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND LICENSEE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
* DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, CLICK ON AN "ACCEPT" BUTTON, OR USE THE PROGRAM; AND
* PROMPTLY RETURN THE UNUSED MEDIA, DOCUMENTATION, AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM IT WAS OBTAINED FOR A REFUND OF THE AMOUNT PAID. IF THE PROGRAM WAS DOWNLOADED, DESTROY ALL COPIES OF THE PROGRAM.
Read Non-IBM Terms
Print
Accept
Decline

Figure 3-15 IBM Spectrum Scale Software License Agreement

4. When installation has completed, the last window of the wizard is displayed. Click **Finish**. See Figure 3-16 on page 124.

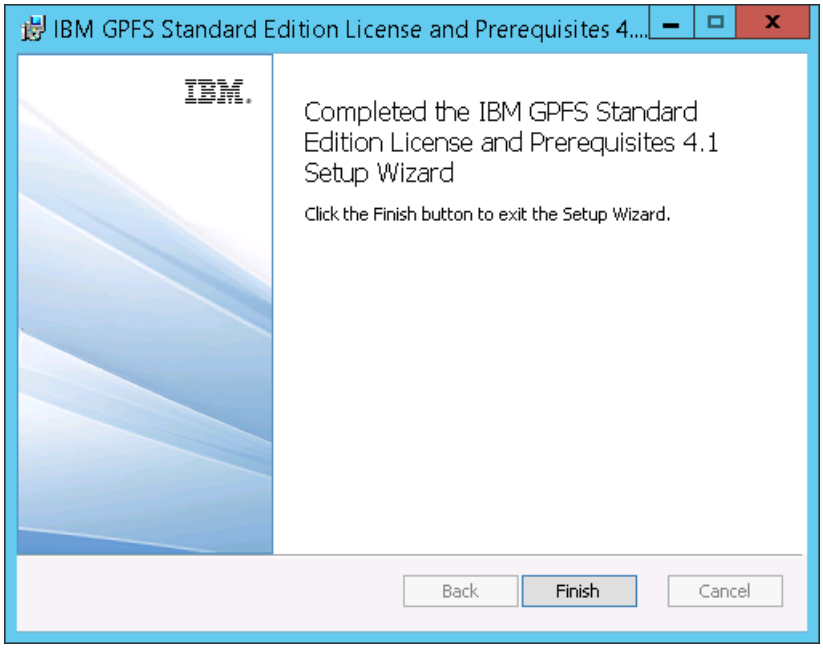

IBM GPFS Standard Edition License and Prerequisites 4....
IBM.
Completed the IBM GPFS Standard Edition License and Prerequisites 4.1 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

Figure 3-16 Final IBM Spectrum Scale license wizard step

5. After the appropriate license is installed in all nodes, proceed to install the IBM Spectrum Scale binaries that match the license just installed in all the nodes of the cluster. First, select and execute the appropriate file, in this scenario, the `gpfs.ext-4.1.0.4-Windows.msi`. See Figure 3-17.

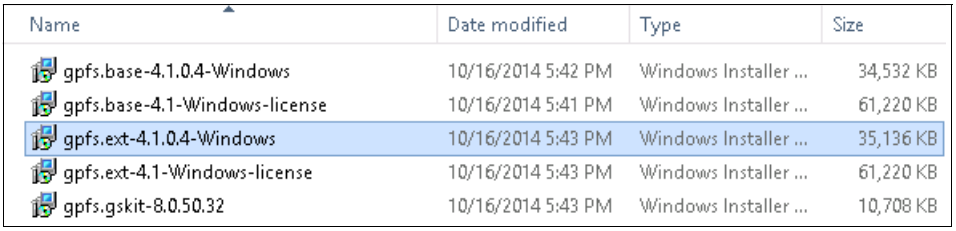


Figure 3-17 IBM Spectrum Scale Standard Edition binaries file

6. Select **Next** on the welcome panel of the IBM Spectrum Scale binaries installation program. See Figure 3-18 on page 125.

IBM GPFS Standard Edition 4.1.0.4 Setup
IBM.
Welcome to the IBM GPFS Standard Edition 4.1.0.4 Setup Wizard
The Setup Wizard will install IBM GPFS Standard Edition 4.1.0.4 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

Figure 3-18 Welcome panel of IBM Spectrum Scale Standard Edition license

7. When all the checks are passed, the installation panel appears. Click **Install**. See Figure 3-19.

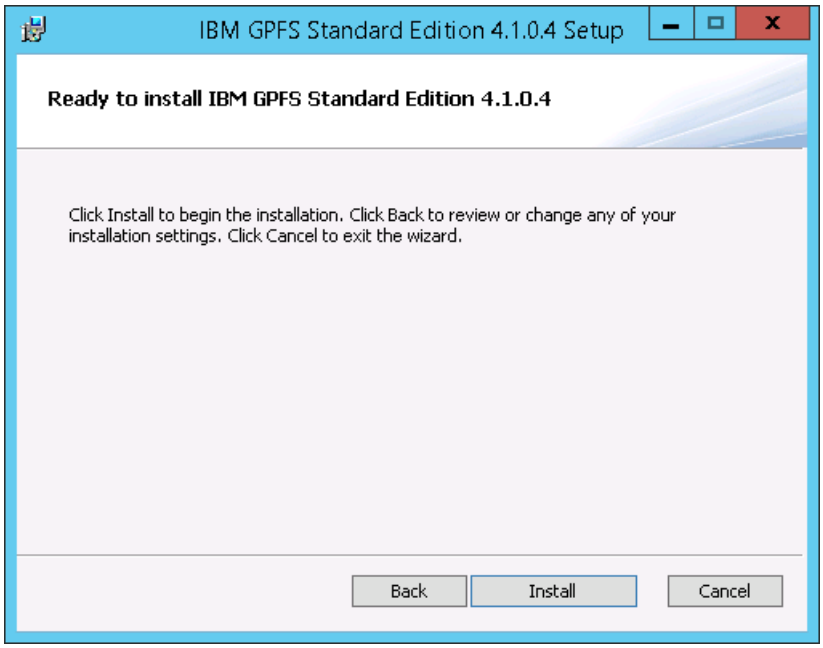

IBM GPFS Standard Edition 4.1.0.4 Setup
Ready to install IBM GPFS Standard Edition 4.1.0.4
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

Figure 3-19 Installation panel

8. When the installation has completed, the last panel of the wizard is displayed. Click **Finish**. See Figure 3-20 on page 126.

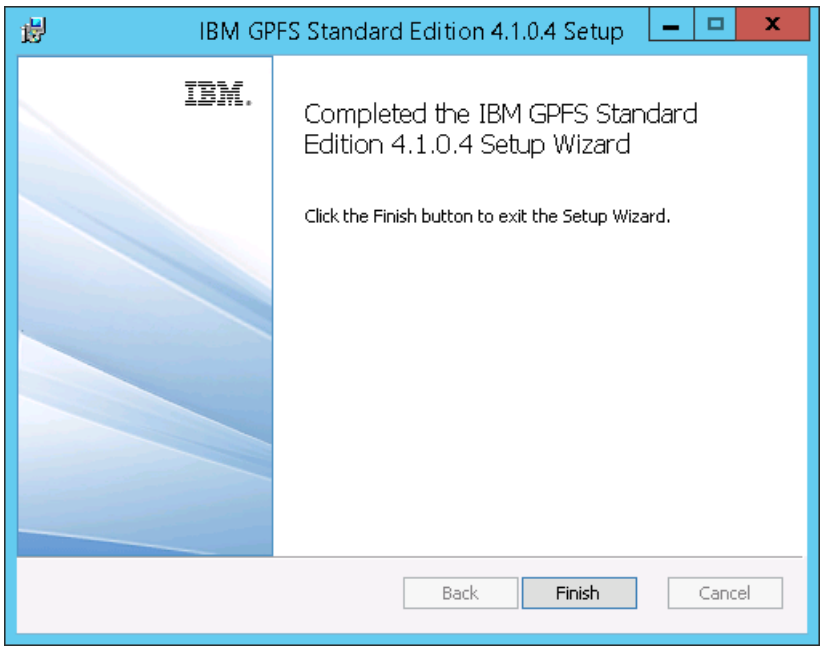

IBM GPFS Standard Edition 4.1.0.4 Setup
IBM.
Completed the IBM GPFS Standard Edition 4.1.0.4 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

Figure 3-20 Final IBM Spectrum Scale binaries wizard step

9. After the IBM Spectrum Scale binaries are installed in all nodes, proceed to install the GSKit in all the nodes of the cluster. First, select and execute the appropriate file, in this scenario, the `gpfs.gskit-8.050.32.msi`. See Figure 3-21.

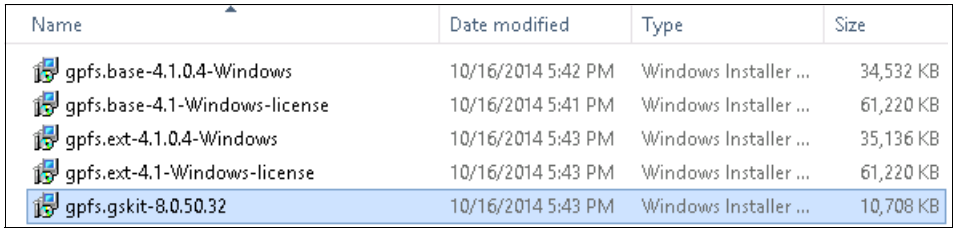


Figure 3-21 GSKit file

10. Select **Next** on the welcome panel of the GSKit installation program. See Figure 3-22 on page 127.

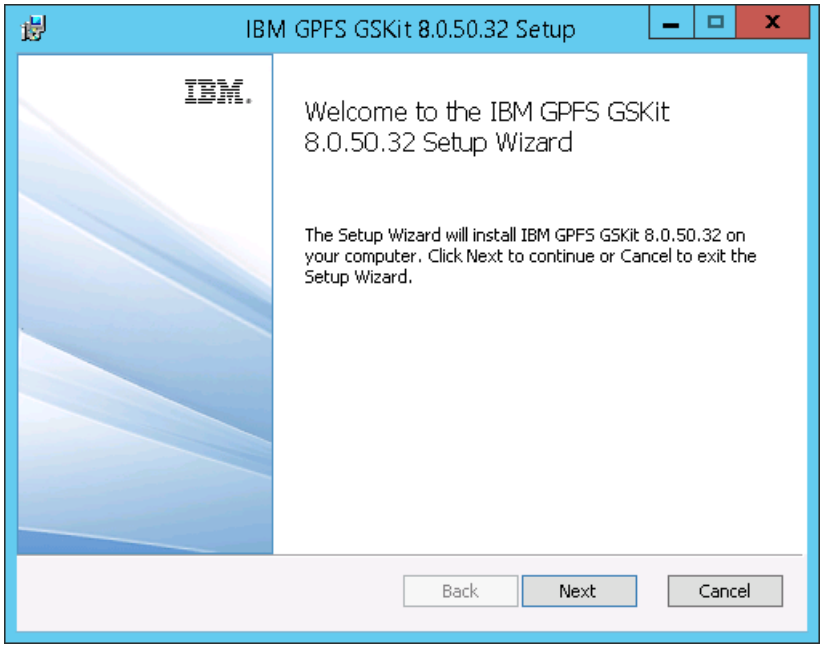

IBM GPFS GSKit 8.0.50.32 Setup
IBM.
Welcome to the IBM GPFS GSKit 8.0.50.32 Setup Wizard
The Setup Wizard will install IBM GPFS GSKit 8.0.50.32 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

Figure 3-22 GSKit welcome wizard panel

11. When all the checks are passed, the installation panel appears. Click **Install**. See Figure 3-23.

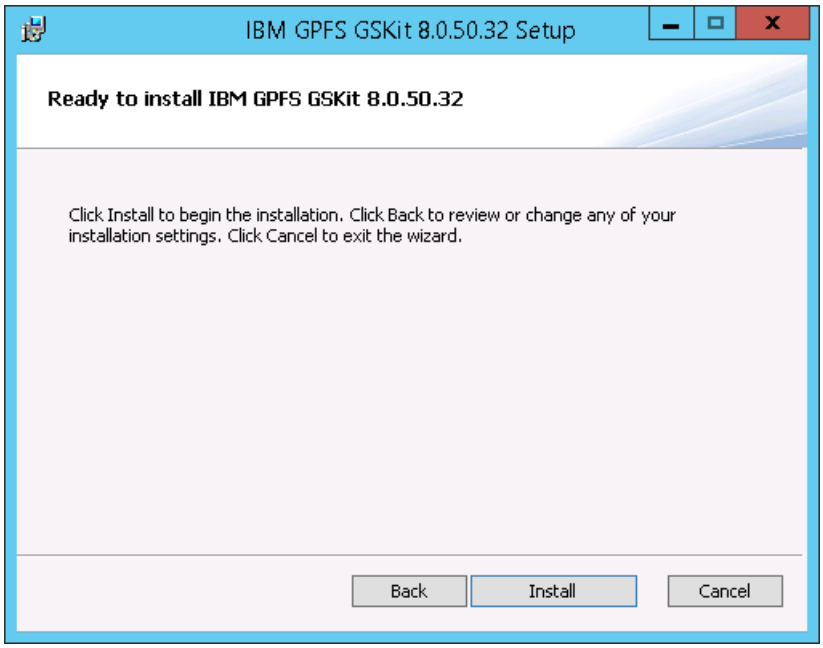

IBM GPFS GSKit 8.0.50.32 Setup
Ready to install IBM GPFS GSKit 8.0.50.32
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

Figure 3-23 GSKit installation wizard window

12. Once the GSKit installation is completed, the wizard shows its last window. Click **Finish**. See Figure 3-24 on page 128.

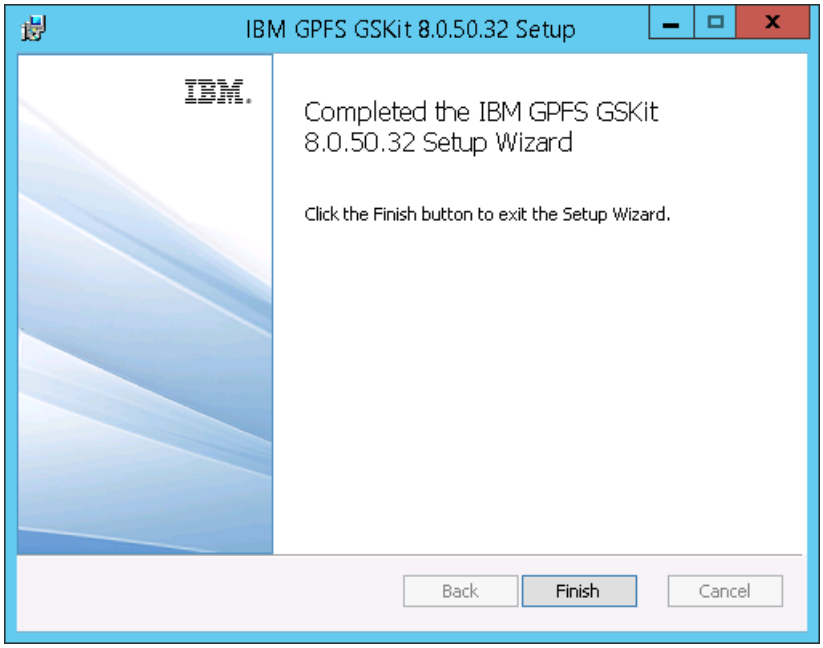


Figure 3-24 GSKit completed wizard panel

13. Once the GSKit is installed in all nodes, reboot all Windows operating system nodes where the GSKit was installed.

3.6.10 Configure IBM Spectrum Scale embedded remote shell

mmwinserv is an IBM Spectrum Scale for Windows service that is needed for the proper functioning of the Spectrum Scale daemon on nodes running Windows operating system. Optionally, the service can be configured to provide a remote execution facility for Spectrum Scale administration commands.

Note: All the Spectrum Scale commands in this scenario run by using the *Spectrum Scale Admin Korn Shell*.

On each node of the cluster, the **mmwinrsh** and **mmwinrcp** programs should be configured.

To configure the mmwinserv daemon, the **mmwinservctl** command is used in both nodes. See Example 3-22.

Note: In this scenario, the **mmwinservctl** command runs with the default security and users.

Example 3-22 Configuring mmwinserv daemon

```
Administrator@olut:~ $ mmwinservctl set

Node name        Service state  Remote shell  Account name
-----------------------------------------------------------
olut             RUNNING        yes           LocalSystem
Administrator@olut:~ $ mmwinservctl enable

Node name        Service state  Remote shell  Account name
```

```
-------------------------------------------------------
olut             RUNNING        yes           LocalSystem
Administrator@olut:~ $ mmwinservctl query

Node name        Service state  Remote shell  Account name
-------------------------------------------------------
olut             RUNNING        yes           LocalSystem
Administrator@olut:~ $

Administrator@palinca ~ $ mmwinservctl set

Node name        Service state  Remote shell  Account name
-------------------------------------------------------
palinca          RUNNING        yes           LocalSystem
Administrator@palinca ~ $ mmwinservctl enable

Node name        Service state  Remote shell  Account name
-------------------------------------------------------
palinca          RUNNING        yes           LocalSystem
Administrator@palinca ~ $ mmwinservctl query

Node name        Service state  Remote shell  Account name
-------------------------------------------------------
palinca          RUNNING        yes           LocalSystem
Administrator@palinca ~ $
```

3.6.11 Create IBM Spectrum Scale cluster

To create the IBM Spectrum Scale cluster in our example, we create a file called `windows.nodes` that has the content as shown in Example 3-23.

Note: From now on in this example, unless explicitly stated, the Spectrum Scale commands are run in only one node of the cluster.

Example 3-23 windows.nodes file

```
olut:quorum-manager
palinca:quorum-manager
```

To create a cluster, run the **mmcrcluster** command as shown in Example 3-24.

Example 3-24 Creating the IBM Spectrum Scale cluster

```
$ mmcrcluster -N windows.nodes -r /usr/lpp/mmfs/bin/mmwinrsh -R /usr/lpp/mmfs/bin/mmwinrcp -C HARDWORK
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: 6027-1254 Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmcrcluster: 6027-1949 Propagating the cluster configuration data to all affected nodes.$
```

Note: By default, the cluster is created using CCR.

After the cluster is created and when it is certain that the correct licenses are provided, change the license of the nodes accordingly. To do so, use the **mmchlicense** command. In this

scenario, both nodes are NSD servers, so run the **mmchlicense** command as shown in Example 3-25.

Example 3-25 Changing the IBM Spectrum Scale license

```
$ mmchlicense server --accept -N olut,palinca

The following nodes will be designated as possessing GPFS server licenses:
        palinca.hardwork.local
        olut.hardwork.local
mmchlicense: Command successfully completed
mmchlicense: 6027-1949 Propagating the cluster configuration data to all affected nodes.
```

Check that the cluster is created by using the **mmlscluster** command as shown in Example 3-26.

Example 3-26 Listing the cluster

```
$ mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         HARDWORK.hardwork.local
  GPFS cluster id:           10543779640873420728
  GPFS UID domain:           HARDWORK.hardwork.local
  Remote shell command:      /usr/lpp/mmfs/bin/mmwinrsh
  Remote file copy command:  /usr/lpp/mmfs/bin/mmwinrcp
  Repository type:           CCR

 Node  Daemon node name            IP address     Admin node name         Designation
-----------------------------------------------------------------------------------
   1   olut.hardwork.local         172.16.20.171  olut.hardwork.local     quorum-manager
   2   palinca.hardwork.local      172.16.20.172  palinca.hardwork.local  quorum-manager
```

3.6.12 Creating NSD disks

In this setup, both nodes are NSD servers, and the storage is shared across both nodes. In this scenario, there are five shared disks between the two nodes. The setup includes IBM Spectrum Virtualize disks with Subsystem Device Driver Device Specific Module (SDDDSM) driver installed. The disks that are used are shown in Example 3-27.

Note: On Windows operating system, the Spectrum Scale NSD servers need to be physical servers.

Example 3-27 Shared disks between the Windows operating system nodes

```
$ pcmpath query device
DEV#:   1  DEVICE NAME: Disk2 Part0  TYPE: 2145       POLICY: OPTIMIZED
SERIAL: 600507680191026C4000000000000190        LUN SIZE:  5.0GB
============================================================================
Path#            Adapter/Hard Disk        State  Mode       Select     Errors
    0 *   Scsi Port2 Bus0/Disk2 Part0     OPEN   NORMAL        490          0

DEV#:   2  DEVICE NAME: Disk3 Part0  TYPE: 2145       POLICY: OPTIMIZED
SERIAL: 600507680191026C4000000000000191        LUN SIZE:  5.0GB
============================================================================
Path#            Adapter/Hard Disk        State  Mode       Select     Errors
    0     Scsi Port2 Bus0/Disk3 Part0     OPEN   NORMAL        526          0
```

```
DEV#:   3  DEVICE NAME: Disk4 Part0  TYPE: 2145       POLICY: OPTIMIZED
SERIAL: 600507680191026C4000000000000192        LUN SIZE:  5.0GB
==========================================================================
Path#            Adapter/Hard Disk        State  Mode       Select     Errors
    0 *   Scsi Port2 Bus0/Disk4 Part0     OPEN   NORMAL        511          0

DEV#:   4  DEVICE NAME: Disk5 Part0  TYPE: 2145       POLICY: OPTIMIZED
SERIAL: 600507680191026C4000000000000193        LUN SIZE:  5.0GB
==========================================================================
Path#            Adapter/Hard Disk        State  Mode       Select     Errors
    0     Scsi Port2 Bus0/Disk5 Part0     OPEN   NORMAL        521          0
```

To create the NSD disks, a file with the NSD definition must be created. In this scenario, only three disks are used, all in the same failure group and in the *system* pool. We use stanza file called `disks.nsd` and its content is shown in Example 3-28.

Example 3-28 disks.nsd file

```
%nsd:
 device=1
 nsd=NSD001
 servers=olut,palinca
 usage=dataAndMetadata
 failureGroup=1
 pool=system

%nsd:
 device=2
 nsd=NSD002
 servers=palinca,olut
 usage=dataAndMetadata
 failureGroup=1
 pool=system

%nsd:
 device=3
 nsd=NSD003
 servers=olut,palinca
 usage=dataAndMetadata
 failureGroup=1
 pool=system
```

Note: Please notice that the order of servers on NSD is different for NSD002. The idea is to have is to spread the load of the NSD across all the servers. Ideally you would like to spread the NSD alternating the primary server on each. The more NSD and servers the more spread the load.

To create the NSD disks, the **mmcrnsd** command must be used with the input file from Example 3-28. Example 3-29 shows how the **mmcrnsd** command is used to create the NSD disks and how the **mmlsnsd** command is used to list the NSD disks.

Example 3-29 disks.nsd

```
$ mmcrnsd -F disks.nsd
mmcrnsd: Processing disk 1
mmcrnsd: Processing disk 2
mmcrnsd: Processing disk 3
mmcrnsd: 6027-1949 Propagating the cluster configuration data to all affected nodes.
```

```
$ mmlsnsd -L -a

 File system   Disk name    NSD volume ID      NSD servers
------------------------------------------------------------------------------------------
 (free disk)   NSD001       14ABAC10544193C3   olut.hardwork.local,palinca.hardwork.local
 (free disk)   NSD002       14ABAC10544193C5   palinca.hardwork.local,olut.hardwork.local
 (free disk)   NSD003       14ABAC10544193C7   olut.hardwork.local,palinca.hardwork.local
```

Regardless of the disk technology used, it is possible to get the Windows operating system disk number from the *Server Manager* program. Select **File and Storage Services**, then select **Volumes**, and select **Disks**. See Figure 3-25 where the already created NSDs are displayed as seen from the *olut* server. This view does not tell if the disks are shared or not with the *palinca* server.

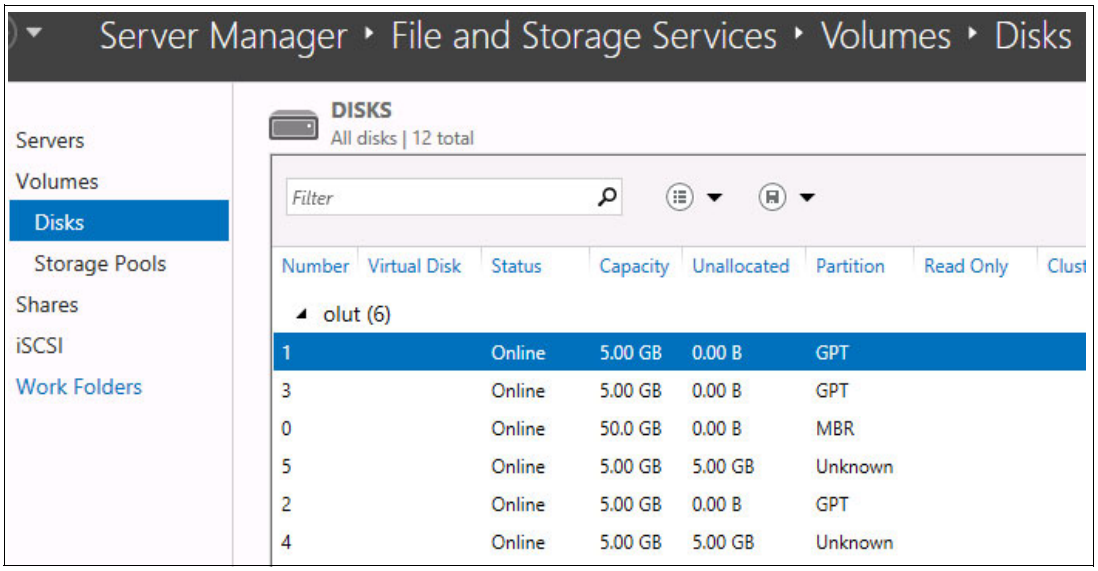


Figure 3-25 Listing Windows operating system disk number

Note: On Windows operating system, Spectrum Scale only creates NSDs from empty disk drives. The **mmcrnsd** command accepts Windows operating system basic disks or unknown/not Initialized disks. The command always reinitializes these disks so that they become basic GPT disks with a single Spectrum Scale partition. NSD data is stored in Spectrum Scale partitions. This allows other operating system components to recognize that the disks are in use.

3.6.13 Start the IBM Spectrum Scale cluster

At this moment, we have a two-node setup cluster with three NSD disks defined. Before creating the Spectrum Scale file system, start the cluster with the **mmstartup** command and list its status with the **mmgetstate** command as shown in Example 3-30.

Example 3-30 Starting and listing state of IBM Spectrum Scale cluster

```
$ mmstartup -a
Fri, Oct 17, 2014  6:16:14 PM: 6027-1642 mmstartup: Starting GPFS ...
$ mmgetstate -L -a

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------
```

```
      1      olut             2       2         2       active      quorum node
      2      palinca          2       2         2       active      quorum node
```

Note: In case you run into problems, check the Spectrum Scale logs, which are in the `/var/adm/ras` directory.

3.6.14 Quorum configuration

The setup now has two nodes and a quorum of two, which means that if any of the two nodes are not active, quorum is not reached. If you are forced to use only two nodes, and need to have the file system up even with one node down, you have to use the tiebreaker disk to uneven the quorum. To configure the tiebreaker disk, use the **`mmchconfig`**, **`mmlsconfig`**, and **`mmgetstate`** commands as shown in Example 3-31.

Example 3-31 Adding tiebreaker disks and listing current quorum status

```
First get the current quorum and cluster status:
$ mmgetstate -L -a -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      olut              2       2          2       active      quorum node
       2      palinca           2       2          2       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:           2
Number of local nodes active in the cluster:      2
Number of remote nodes joined in this cluster:    0
Number of quorum nodes defined in the cluster:    2
Number of quorum nodes active in the cluster:     2
Quorum = 2, Quorum achieved

Quorum is 2 and we have both nodes up, so quorum is achieved.

In this setup all three NSD disks will be added as tie breakers:

$ mmchconfig tiebreakerDisks="NSD001;NSD002;NSD003"
mmchconfig: Command successfully completed
mmchconfig: 6027-1949 Propagating the cluster configuration data to all affected nodes.
$ mmgetstate -L -a -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      olut              1       2          2       active      quorum node
       2      palinca           1       2          2       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:           2
Number of local nodes active in the cluster:      2
Number of remote nodes joined in this cluster:    0
Number of quorum nodes defined in the cluster:    2
Number of quorum nodes active in the cluster:     2
Quorum = 1*, Quorum achieved

Now quorum is 1 and we have achieved 2 nodes up and quorum.
```

```
On the cluster configuration see the NSD disks that are tiebreakers:
$ mmlsconfig
Configuration data for cluster HARDWORK.hardwork.local:
------------------------------------------------------
clusterName HARDWORK.hardwork.local
clusterId 10543779640873420728
autoload no
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks NSD001;NSD002;NSD003
adminMode central

File systems in cluster HARDWORK.hardwork.local:
------------------------------------------------
(none)
```

3.6.15 Start IBM Spectrum Scale at boot

In this setup, we want the Spectrum Scale cluster to come up at boot time. In other scenarios, this might not be the wanted setup. To enable Spectrum Scale to start at boot, use the **mmchconfig** command. To list the current value of the **autoload** attribute, use the **mmlsconfig** command as shown in Example 3-32.

Example 3-32 Configuring Spectrum Scale to start at boot

```
$ mmchconfig autoload=yes

mmchconfig: Command successfully completed
mmchconfig: 6027-1949 Propagating the cluster configuration data to all affected nodes.
$ mmlsconfig
Configuration data for cluster HARDWORK.hardwork.local:
------------------------------------------------------
clusterName HARDWORK.hardwork.local
clusterId 10543779640873420728
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks NSD001;NSD002;NSD003
autoload yes
adminMode central

File systems in cluster HARDWORK.hardwork.local:
------------------------------------------------
(none)
```

3.6.16 Create the IBM Spectrum Scale file system

In this example, only the simplest file system is created using defaults for all parameters and the input file from Example 3-28 on page 131. To create an IBM Spectrum Scale file system, the **mmcrfs** command must be used. Example 3-33 on page 135 shows how the cluster is started in all nodes with the **mmstartup** command, and then how the IBM Spectrum Scale file system is created and listed.

Then, while running Spectrum Scale on Linux nodes, use the **-t** option for the **mmcrfs** command, which allows to designate a Windows operating system drive letter. In

Example 3-33 on page 135, we assigned the drive letter *G* for the new IBM Spectrum Scale file system.

Example 3-33 Creating an IBM Spectrum Scale file system

```
$ mmcrfs BAGUETTE -F disks.nsd -t G

GPFS: 6027-531 The following disks of BAGUETTE will be formatted on node palinca:
    NSD001: size 5103 MB
    NSD002: size 5103 MB
    NSD003: size 5103 MB
GPFS: 6027-540 Formatting file system ...
GPFS: 6027-535 Disks up to size 103 GB can be added to storage pool system.
Creating Inode File
  32 % complete on Mon Oct 20 10:01:48 2014
  51 % complete on Mon Oct 20 10:01:53 2014
  70 % complete on Mon Oct 20 10:01:58 2014
  94 % complete on Mon Oct 20 10:02:03 2014
 100 % complete on Mon Oct 20 10:02:04 2014
Creating Allocation Maps
Creating Log Files
  53 % complete on Mon Oct 20 10:02:12 2014
 100 % complete on Mon Oct 20 10:02:15 2014
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
GPFS: 6027-572 Completed creation of file system /dev/BAGUETTE.
mmcrfs: 6027-1949 Propagating the cluster configuration data to all affected nodes.
```

To mount the file system, use the **mmmount** command. To check if the file system is mounted, either use operating system commands or the **mmlsmount** command. See Example 3-34.

Example 3-34 Mounting and listing the mount status of the file system

```
$ mmmount BAGUETTE -a
Mon, Oct 20, 2014 10:17:29 AM: 6027-1623 mmmount: Mounting file systems ...
$ mmlsmount BAGUETTE -L

File system BAGUETTE is mounted on 2 nodes:
  172.16.20.172   palinca
  172.16.20.171   olut
```

Now you can use the file system normally as any other hard disk drive in Windows operating systems. All nodes that mount the file system, have read/write access to the drive.

Note: This scenario intends to show a simple case of two Windows operating system nodes in a Spectrum Scale cluster. The scenario uses default settings for the cluster at file system creation. This scenario is not intended to be the most feasible way to set up a two-node cluster; this scenario is used to show the creation of a Windows operating system cluster.

3.6.17 Adding a Linux node to the Windows operating system cluster

Windows operating system nodes have certain constraints on which operations can be done from the Windows system nodes into an IBM Spectrum Scale file system (refer to 2.4.4, “IBM Spectrum Scale on Windows operating systems” on page 61). It is possible to add a non

Windows system node to the cluster that can be used to overcome the situation. In this section, a Linux on x86 Spectrum Scale client node is added to the Windows operating system-only cluster. See Figure 3-26 on page 136.

Note: Check that TCP ports 22 and 1191 are allowed both ways in all nodes. In this scenario, the Linux firewall was also disabled as with the Windows operating system nodes.

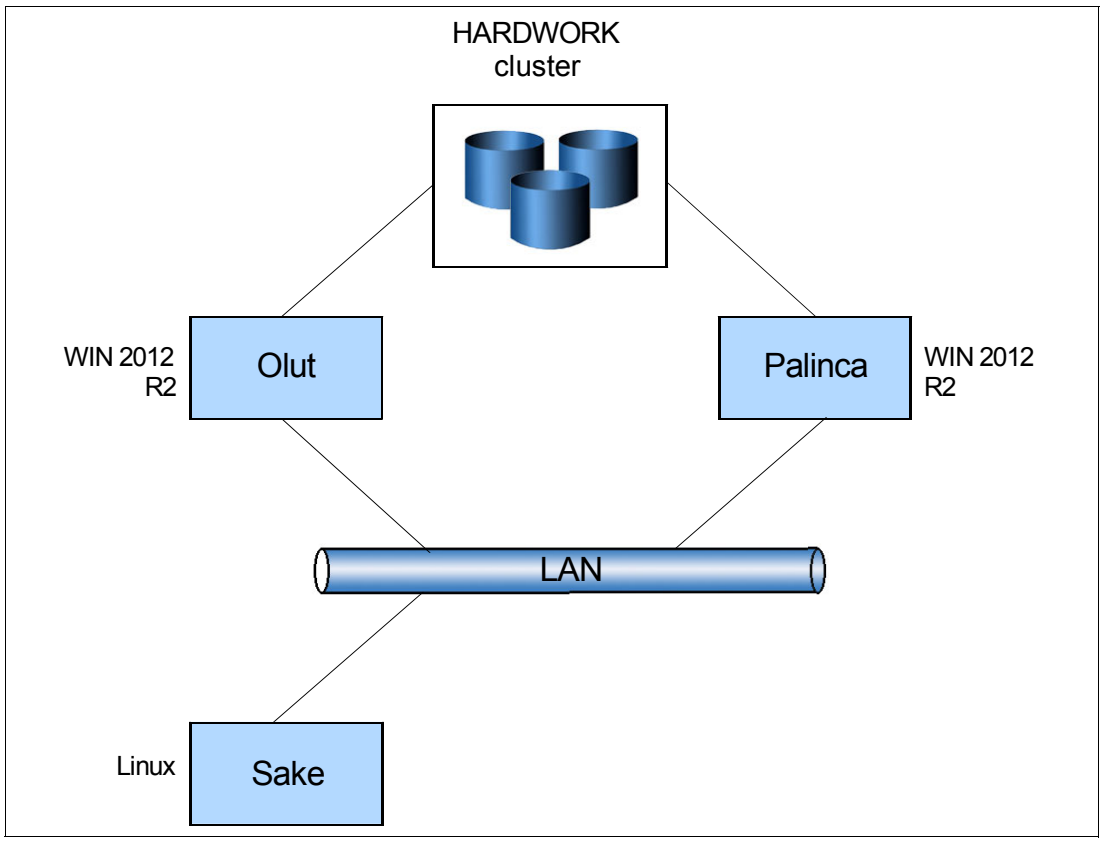


Figure 3-26 Windows operating system and Linux cluster

Install IBM Spectrum Scale

All the nodes in the cluster should have the same IBM Spectrum Scale Edition. In this scenario, we install IBM Spectrum Scale 4.1 TL1 Standard Edition for Linux on x86_64. The node is a Spectrum Scale NSD client. On each node, run the **rpm** command as shown in Example 3-35.

Example 3-35 Installing IBM Spectrum Scale Standard Edition on Linux

```
# rpm -vih gpfs.base*.x86_64.rpm gpfs.ext-*.x86_64.rpm gpfs.docs*.noarch.rpm gpfs.gskit-*.x86_64.rpm
gpfs.libsrc-*.noarch.rpm gpfs.msg.en_US-*.noarch.rpm gpfs.src-*.noarch.rpm gpfs.gpl*.noarch.rpm
Preparing...                ########################################### [100%]
   1:gpfs.base              ########################################### [ 13%]
   2:gpfs.ext               ########################################### [ 25%]
   3:gpfs.gpl               ########################################### [ 38%]
   4:gpfs.src               ########################################### [ 50%]
   5:gpfs.msg.en_US         ########################################### [ 63%]
   6:gpfs.libsrc            ########################################### [ 75%]
   7:gpfs.gskit             ########################################### [ 88%]
```

Configure auxiliary tools

In this section, the needed configurations and tools before adding the Linux node to the Windows operating system cluster only are explained.

Configuring name resolution

"File /etc/hosts" on page 100 explains the usage of file resolution in a Spectrum Scale cluster. In this scenario, we use the Domain Name Server (DNS) that contains the Windows operating system nodes and add our new node to the Windows DNS records of the zone. Then, on the Linux node, configure it to use the DNS server. In this case, it is done by adding the lines to the `ifcfg-eth0` file.

Identity Management for UNIX

IBM Spectrum Scale can exploit a Windows server feature called *Identity Management for UNIX* (IMU) to provide consistent identities among all nodes in a cluster.

IBM Spectrum Scale expects that all Windows operating system nodes in a cluster are members of the same active directory domain. This gives domain users a consistent identity and consistent file access rights independent of the system they are using.

This is an optional component to keep the file ownership consistency. For IMU installation and configuration information, see the topic about Identity Management on Windows operating system in the *IBM Spectrum Scale: Advanced Administration Guide*:

`http://ibm.co/1HseH36`

Creating the Spectrum Scale administrative account

Spectrum Scale uses an administrative account in the active directory domain named *root* in order to interoperate with UNIX nodes in the cluster. Create the administrative account as follows:

1. Stop the Spectrum Scale cluster.
2. Create a domain user with the logon name root.
3. Add user root to the domain admins group or to the local administrators group on each Windows operating system node.
4. In root `Properties/Profile/Home/LocalPath`, define a HOME directory such as `C:\Users\root\home` that does not include spaces in the path name and is not the same as the profile path. See Figure 3-27 on page 138.

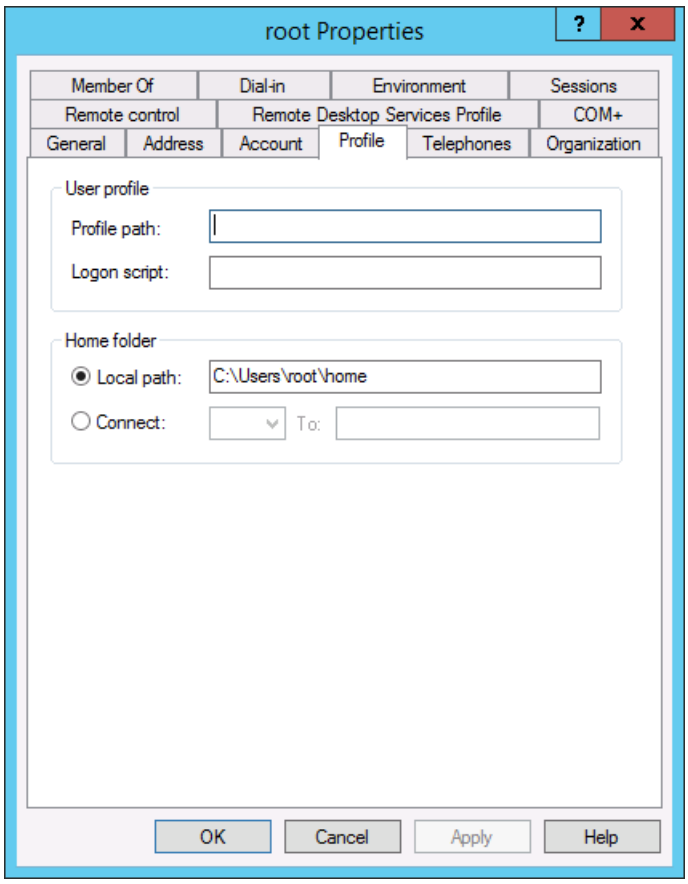


Figure 3-27 root user Properties

5. Give root the right to log on as a service:
   - Open Group Policy Management (available under the Administrative Tools). It can be opened by pressing the Windows operating system logo key + R to open the RUN dialog box. Type gpmc.msc in the text box, and then click **OK** or press Enter. See Figure 3-28.

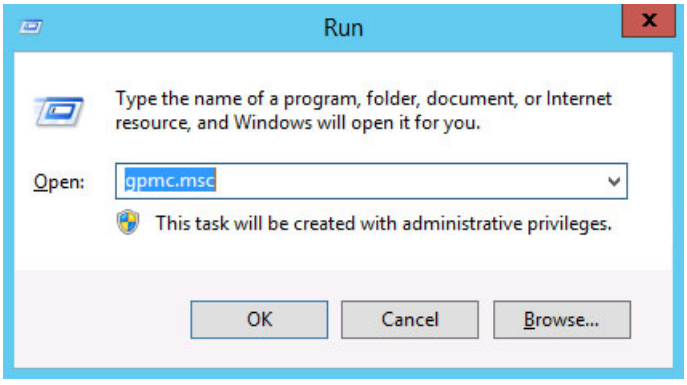

Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open:
gpmc.msc
This task will be created with administrative privileges.
OK
Cancel
Browse...

Figure 3-28 Running GPMC

   - In the console tree, expand **Forest name** → **Domains** → **Domain name** → **Group Policy Objects**. Right-click **Default Domain Policy** and select **Edit**. See Figure 3-29 on page 139.

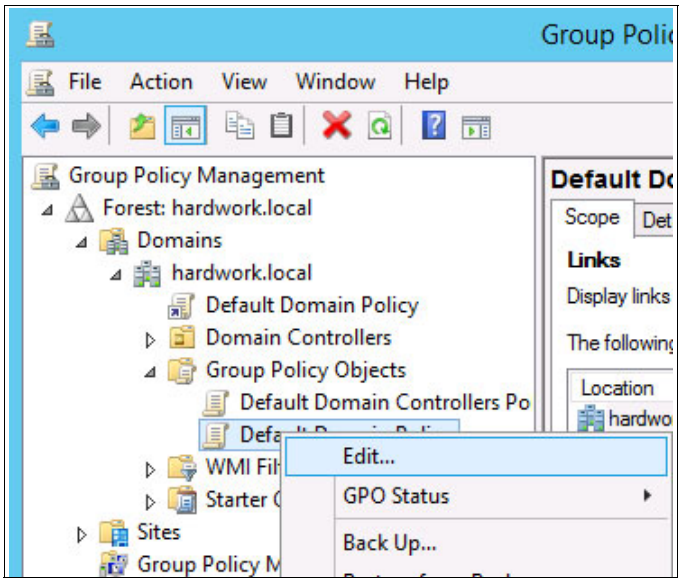


Figure 3-29 Edit Group Default Policy Objects

– In the console tree of the Group Policy Management Editor, expand down to **Computer Configuration** → **Policies** → **Windows Settings** → **Security Settings** → **Local Policies** → **User Rights Assignment**. See Figure 3-30.

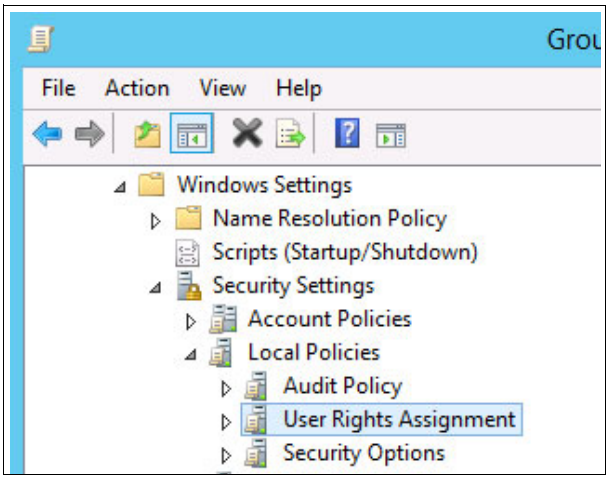


Figure 3-30 User rights assignment

– Double-click Log on as a service policy.

– Check Define these policy settings if necessary. Use Add User or Group to include the `DomainName\root` account in the policy, then click **OK**. See Figure 3-31 on page 140.

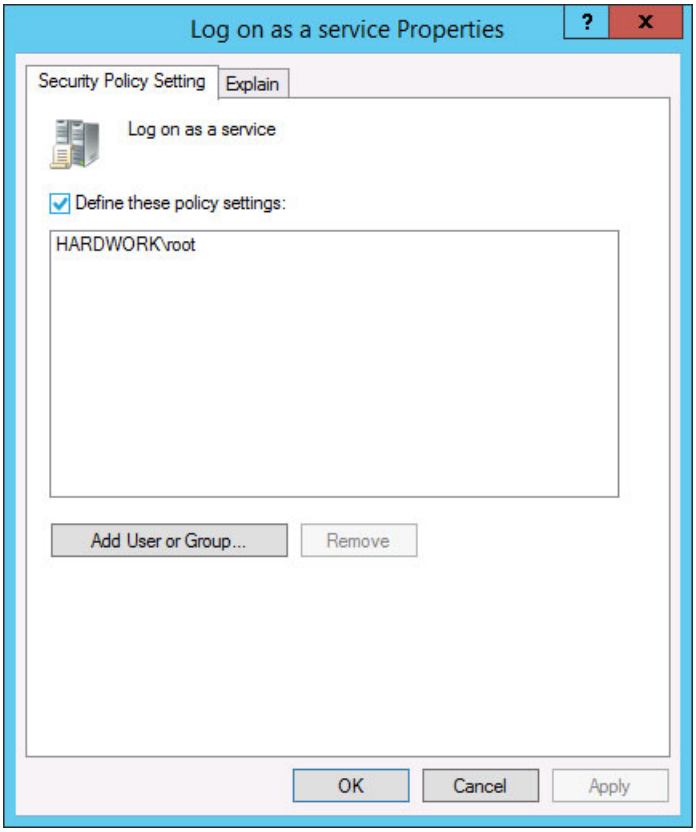


Figure 3-31 Log on as a service root user

6. Use the Spectrum Scale **mmwinservctl** command to set and maintain the Spectrum Scale Administration service's logon account. The **mmwinservctl** command must be run on a Windows operating system node.

Example 3-36 Changing mmwinserv user to root from Administrator

```
$ mmwinservctl set -N olut,palinca --account HARDWORK/root --password "Redb00k;" --remote-shell no

Node name       Service state  Remote shell  Account name
-----------------------------------------------------------
olut            RUNNING        no            HARDWORK\root
palinca         RUNNING        no            HARDWORK\root
```

7. Reboot the Windows operating system nodes.

Note: In Example 3-36, the `HARDWORK\root` user has a non-expiring password. If it needs to be changed, update the password with the **mmwinservctl** command:

```
$ mmwinservctl set -N all --password mynewpwd
```

Remote shell

OpenSSH is not a requirement when using the Windows operating system-only cluster. However, when adding a non Windows operating system node, OpenSSH should be used.

Note: In this scenario, the **`PermitRootLogin yes`** parameter is added to the `/etc/sshd_config` file. If your setup uses other users than administrator, there is no need to add this to the setup.

If not done in 3.6.8, “Installing Cygwin” on page 121, install OpenSSH on both Windows operating system nodes. To do so, perform the following steps on each node:

- Open a Cygwin shell window.
- Run the **`ssh-host-config`** command to configure the SSH server on the Windows operating system nodes.
- Either run the **`cygrunsrv -S sshd`** command or reboot the Windows operating system node.
- Run the **`ssh-user-config`** command.

Note: It is recommended that you choose **no** when installing OpenSSH to this query:

```
*** Query: Should privilege separation be used? (yes/no) no
```

Example 3-37 shows how to do this on the Cygwin terminal.

Example 3-37 Configuring and starting SSH server on Windows operating system nodes

```
$ ssh-host-config

*** Info: Generating missing SSH host keys
ssh-keygen: generating new host keys: RSA1 RSA DSA ECDSA ED25519
*** Info: Creating default /etc/ssh_config file
*** Info: Creating default /etc/sshd_config file

*** Info: StrictModes is set to 'yes' by default.
*** Info: This is the recommended setting, but it requires that the POSIX
*** Info: permissions of the user's home directory, the user's .ssh
*** Info: directory, and the user's ssh key files are tight so that
*** Info: only the user has write permissions.
*** Info: On the other hand, StrictModes don't work well with default
*** Info: Windows permissions of a home directory mounted with the
*** Info: 'noacl' option, and they don't work at all if the home
*** Info: directory is on a FAT or FAT32 partition.
*** Query: Should StrictModes be used? (yes/no) yes

*** Info: Privilege separation is set to 'sandbox' by default since
*** Info: OpenSSH 6.1.  This is unsupported by Cygwin and has to be set
*** Info: to 'yes' or 'no'.
*** Info: However, using privilege separation requires a non-privileged account
*** Info: called 'sshd'.
*** Info: For more info on privilege separation read /usr/share/doc/openssh/README.privsep.
*** Query: Should privilege separation be used? (yes/no) no
*** Info: Updating /etc/sshd_config file

*** Query: Do you want to install sshd as a service?
*** Query: (Say "no" if it is already installed as a service) (yes/no) yes
*** Query: Enter the value of CYGWIN for the daemon: []
*** Info: On Windows Server 2003, Windows Vista, and above, the
```

```
*** Info: SYSTEM account cannot setuid to other users -- a capability
*** Info: sshd requires.  You need to have or to create a privileged
*** Info: account.  This script will help you do so.

*** Info: You appear to be running Windows XP 64bit, Windows 2003 Server,
*** Info: or later.  On these systems, it's not possible to use the LocalSystem
*** Info: account for services that can change the user id without an
*** Info: explicit password (such as passwordless logins [e.g. public key
*** Info: authentication] via sshd).

*** Info: If you want to enable that functionality, it's required to create
*** Info: a new account with special privileges (unless a similar account
*** Info: already exists). This account is then used to run these special
*** Info: servers.

*** Info: Note that creating a new user requires that the current account
*** Info: have Administrator privileges itself.

*** Info: No privileged account could be found.

*** Info: This script plans to use 'cyg_server'.
*** Info: 'cyg_server' will only be used by registered services.
*** Query: Do you want to use a different name? (yes/no) no
*** Query: Create new privileged user account 'cyg_server'? (yes/no) yes
*** Info: Please enter a password for new user cyg_server.  Please be sure
*** Info: that this password matches the password rules given on your system.
*** Info: Entering no password will exit the configuration.
*** Query: Please enter the password:
*** Query: Reenter:

*** Info: User 'cyg_server' has been created with password 'Redb00k;'.
*** Info: If you change the password, please remember also to change the
*** Info: password for the installed services which use (or will soon use)
*** Info: the 'cyg_server' account.

*** Info: Also keep in mind that the user 'cyg_server' needs read permissions
*** Info: on all users' relevant files for the services running as 'cyg_server'.
*** Info: In particular, for the sshd server all users' .ssh/authorized_keys
*** Info: files must have appropriate permissions to allow public key
*** Info: authentication. (Re-)running ssh-user-config for each user will set
*** Info: these permissions correctly. [Similar restrictions apply, for
*** Info: instance, for .rhosts files if the rshd server is running, etc].


*** Info: The sshd service has been installed under the 'cyg_server'
*** Info: account.  To start the service now, call `net start sshd' or
*** Info: `cygrunsrv -S sshd'.  Otherwise, it will start automatically
*** Info: after the next reboot.

*** Info: Host configuration finished. Have fun!

$ cygrunsrv.exe -S sshd
$ ssh-user-config -y
*** Query: Shall I create a SSH2 RSA identity file for you? (yes/no) yes
*** Info: Generating /home/Administrator/.ssh/id_rsa
```

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
*** Query: Do you want to use this identity to login to this machine? (yes/no) yes
*** Info: Adding to /home/Administrator/.ssh/authorized_keys
*** Query: Shall I create a SSH2 ECDSA identity file for you? (yes/no) yes
*** Info: Generating /home/Administrator/.ssh/id_ecdsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
*** Query: Do you want to use this identity to login to this machine? (yes/no) yes
*** Info: Adding to /home/Administrator/.ssh/authorized_keys
*** Query: Shall I create a (deprecated) SSH1 RSA identity file for you? (yes/no) yes
*** Info: Generating /home/Administrator/.ssh/identity
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
*** Query: Do you want to use this identity to login to this machine? (yes/no) yes
*** Info: Adding to /home/Administrator/.ssh/authorized_keys

*** Info: Configuration finished. Have fun!
```

Then, exchange the OpenSSH keys between the two Windows servers by using the following steps:

1. Open Cygwin terminal.
2. With the **scp** command, copy *id_rsa.pub* to the *palinca* server from the *olut* server.
3. Merge with the **cat** command on the *palinca* server an `authorized_files` file that includes both nodes OpenSSH public keys.
4. With the **scp** command, copy *authorized_file* to the *olut* server from the *palinca* server.

Example 3-38 shows how to do this setup.

Example 3-38 Exchanging the OpenSSH public keys between the two Windows system nodes

```
Administrator@olut ~/.ssh$ scp id_rsa.pub palinca:$PWD/id_rsa.pub.olut
The authenticity of host 'palinca (172.16.20.172)' can't be established.
ECDSA key fingerprint is 75:94:c8:24:fb:14:f1:90:74:0d:23:35:a0:35:1f:0a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'palinca,172.16.20.172' (ECDSA) to the list of known hosts.
Administrator@palinca's password:
id_rsa.pub                                                    100%  400     0.4KB/s
00:00

Administrator@palinca ~$ cd .ssh

Administrator@palinca ~/.ssh$ ls
authorized_keys  id_ecdsa      id_rsa.pub       identity.pub
id_dsa           id_ecdsa.pub  id_rsa.pub.olut  known_hosts
id_dsa.pub       id_rsa        identity

Administrator@palinca ~/.ssh$ cat id_rsa.pub.olut >> authorized_keys

Administrator@palinca ~/.ssh$ scp authorized_keys olut:$PWD
The authenticity of host 'olut (172.16.20.171)' can't be established.
ECDSA key fingerprint is 09:b1:4b:9f:cd:58:65:c6:48:5b:0e:b4:13:c8:84:cd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'olut,172.16.20.171' (ECDSA) to the list of known hosts.
```

```
Administrator@olut's password:
authorized_keys                               100% 2859     2.8KB/s   00:00
```

Note: If you receive setgeid or setgid errors for ssh on the Windows event log, and ssh connections are dropped, run the following commands to synchronize Windows operating system and Cygwin user and groups:

```
$ mkgroup -l -d > /etc/group
$ mkpasswd.exe -l -d > /etc/passwd
```

Then, restart sshd.

Check that SSH passwordless works between both nodes and themselves. A way to do so is to run the **date** command via ssh on both nodes. Example 3-39 shows what can be run on each node to ensure that ssh connections are working.

Example 3-39 Checking SSH connectivity and time on a Windows operating system via Cygwin

```
$ ssh olut.hardwork.local date && ssh palinca.hardwork.local date
Wed Oct 22 15:56:12 EDT 2014
Wed Oct 22 15:56:12 EDT 2014
```

When the SSH connectivity is working between the nodes, the Spectrum Scale cluster needs to be reconfigured to use OpenSSH **ssh** and **scp** commands. To do so, follow these steps from one node of the cluster:

1. Open a Spectrum Scale Admin Korn shell.
2. Unmount the Spectrum Scale file system from all nodes with the **mmumount** command.
3. Stop the cluster in all nodes with the **mmshutdown** command.
4. Change the Spectrum Scale configuration with the **mmchconfig** command to use OpenSSH.
5. Check the configuration with the **mmlscluster** command.
6. Start the Spectrum Scale cluster in all nodes with the **mmstartup** command.
7. Check Spectrum Scale cluster status with the **mmgetstate** command.
8. Mount the Spectrum Scale file system in all nodes with the **mmmount** command.

Example 3-40 shows how this reconfiguring can be set up.

Example 3-40 Reconfiguring Windows operating system cluster to use OpenSSH

```
$ mmumount BAGUETTE -a
Wed, Oct 22, 2014  4:05:15 PM: 6027-1674 mmumount: Unmounting file systems ...
$ mmshutdown -a
Wed, Oct 22, 2014  4:05:44 PM: 6027-1341 mmshutdown: Starting force unmount of GPFS file systems
Wed, Oct 22, 2014  4:05:54 PM: 6027-1344 mmshutdown: Shutting down GPFS daemons
Wed, Oct 22, 2014  4:06:27 PM: 6027-1345 mmshutdown: Finished
$ mmchcluster -r /usr/bin/ssh -R /usr/bin/scp
mmsetrcmd: Command successfully completed
mmsetrcmd: 6027-1949 Propagating the cluster configuration data to all affected nodes.
$ mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         HARDWORK.hardwork.local
  GPFS cluster id:           10543779640873750826
  GPFS UID domain:           HARDWORK.hardwork.local
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR
```

```
 Node  Daemon node name            IP address     Admin node name         Designation
-----------------------------------------------------------------------------------------
   1   olut.hardwork.local         172.16.20.171  olut.hardwork.local     quorum-manager
   2   palinca.hardwork.local      172.16.20.172  palinca.hardwork.local  quorum-manager

$ mmstartup -a
Wed, Oct 22, 2014  4:09:36 PM: 6027-1642 mmstartup: Starting GPFS ...
$ mmgetstate -a -L -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
-----------------------------------------------------------------------------------
       1      olut              1        2          2       active      quorum node
       2      palinca           1        2          2       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:           2
Number of local nodes active in the cluster:      2
Number of remote nodes joined in this cluster:    0
Number of quorum nodes defined in the cluster:    2
Number of quorum nodes active in the cluster:     2
Quorum = 1*, Quorum achieved

$ mmmount BAGUETTE
Wed, Oct 22, 2014  4:11:38 PM: 6027-1623 mmmount: Mounting file systems ...
```

When OpenSSH is in use in the Spectrum Scale cluster, it is possible to add the Linux node to the Spectrum Scale cluster. To do so follow these steps:

1. Create the SSH key on the Linux node with the **ssh-keygen** command.
2. Propagate the OpenSSH key to the Windows system nodes `authorized_keys` file.
3. Copy to the Linux node a current `authorized_keys` file that contains all three nodes' public keys.

In Example 3-41, the keys are generated with the **ssh-keygen** command, then propagated across all nodes with the **scp** command. In Example 3-41, all the commands run from the Linux node *sake*.

Example 3-41 Generating the Linux OpenSSH keys and getting all public keys in all nodes

```
[root@sake ~]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.

ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
c9:93:09:99:93:5c:d0:81:39:c3:ec:2a:41:aa:24:d4 root@sake
The key's randomart image is:
+--[ DSA 1024]----+
|  .  o.=o.       |
| ..E .B=.        |
|.o   .Bo         |
|o..   .+ +       |
|+  . .  S        |
|. . .    .       |
```

```
|   .            |
|                |
|                |
+----------------+
[root@sake ~]# cd .ssh
[root@sake .ssh]# scp Administrator@olut:/home/Administrator/.ssh/authorized_keys authorized_keys_windows
Administrator@olut's password:
authorized_keys                                                100% 2859     2.8KB/s   00:00
[root@sake .ssh]# scp Administrator@olut:/home/Administrator/.ssh/authorized_keys authorized_keys_windows
The authenticity of host 'olut (172.16.20.171)' can't be established.
RSA key fingerprint is be:84:4a:a1:e1:b9:89:35:df:66:5a:1d:c7:6e:c0:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'olut,172.16.20.171' (RSA) to the list of known hosts.
Administrator@olut's password:
authorized_keys                                                100% 2859     2.8KB/s   00:00
[root@sake .ssh]# cat id_dsa.pub >> authorized_keys_windows
[root@sake .ssh]# cat authorized_keys_windows >> authorized_keys
[root@sake .ssh]# scp authorized_keys Administrator@olut:/home/Administrator/.ssh/authorized_keys
Administrator@olut's password:
authorized_keys                                                100% 4083     4.0KB/s   00:00
[root@sake .ssh]# scp authorized_keys Administrator@palinca:/home/Administrator/.ssh/authorized_keys
The authenticity of host 'palinca (172.16.20.172)' can't be established.
RSA key fingerprint is ed:eb:98:e6:c6:be:84:de:18:e7:7c:f3:80:ff:55:19.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'palinca,172.16.20.172' (RSA) to the list of known hosts.
Administrator@palinca's password:
authorized_keys                                                100% 4083     4.0KB/s   00:00
```

NTP

Nodes need to have the same time. The easiest way to achieve this is to have the nodes using the same NTP servers, which in our case, are the AD servers. Linux includes an NTP client implementation. Check that all three nodes have the same time with the **`date`** command in combination with SSH. Example 3-42 shows how to configure NTP on each node.

Example 3-42 Checking Linux and Windows operating system nodes time

```
[root@sake ~]# ssh Administrator@olut date; ssh Administrator@palinca date; ssh sake date
Wed Oct 22 16:36:12 EDT 2014
Wed Oct 22 16:36:12 EDT 2014
Wed Oct 22 16:36:12 EDT 2014
```

Compile and install IBM Spectrum Scale GNU GPL portability layer

When IBM Spectrum Scale runs on Linux nodes, it needs a portability layer to be able to run. In our environment, we used the new **`mmbuildgpl`** tool provided by IBM Spectrum Scale 4.1 TL1 as described in “Using the new mmbuildgpl tool” on page 103.

Adding the Linux node to the cluster

In this scenario, the default usernames for each OS are used. Windows operating system uses *Administrator* and Linux *root*. For the scenario described, a homogeneous approach needs to take place. In this case, on each Windows operating system nodes, with Cygwin the `/etc/passswd` entry for *Administrator* is renamed as *root* user.

Note: In other scenarios, a common made-up superuser for all platforms could be used. For more information, see the IBM Spectrum Scale Concepts, Planning, and Installation documentation:

http://ibm.co/1yKltMr

At this stage, it is possible to add the Linux node into our Windows operating system-only cluster. It is done by using the **mmaddnode** command from any node that belongs to the cluster with sufficient rights to run Spectrum Scale administrative commands. In our scenario, either of the Windows operating system nodes can do so as shown in Example 3-43.

Example 3-43 Adding Linux node to the cluster

```
$ mmaddnode -N sake
Wed, Oct 22, 2014  4:55:09 PM: 6027-1664 mmaddnode: Processing node sake.hardwork.local
mmaddnode: Command successfully completed
mmaddnode: 6027-1254 Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmaddnode: 6027-1949 Propagating the cluster configuration data to all affected nodes.
```

After the node is in the cluster, the appropriate license for the new node needs to be configured. This is done with the **mmchlicense** command as shown in Example 3-44.

Example 3-44 Changing the license of the Linux node

```
# mmchlicense client -N sake

The following nodes will be designated as possessing GPFS client licenses:
        sake.hardwork.local
Please confirm that you accept the terms of the GPFS client Licensing Agreement.
The full text can be found at www.ibm.com/software/sla
Enter "yes" or "no": yes
mmchlicense: Command successfully completed
mmchlicense: 6027-1949 Propagating the cluster configuration data to all affected nodes.
```

It is possible to list the nodes in the cluster with the **mmlscluster** command. In Example 3-45, the command is run from the Linux node.

Example 3-45 mmlscluster from the Linux node

```
# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         HARDWORK.hardwork.local
  GPFS cluster id:           10543779640873750826
  GPFS UID domain:           HARDWORK.hardwork.local
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name     IP address     Admin node name         Designation
--------------------------------------------------------------------------------
   1   olut.hardwork.local  172.16.20.171  olut.hardwork.local     quorum-manager
   2   palinca.hardwork.local  172.16.20.172  palinca.hardwork.local  quorum-manager
   3   sake.hardwork.local  172.16.20.170  sake.hardwork.local
```

To mount the Spectrum Scale file system in the Linux node, use the **mmmount** command. Then, use the **mmlsmount** command to check the GFPS file system mount status. See Example 3-46 on page 148.

Example 3-46 Mounting the Spectrum Scale file system in the Linux node

```
[root@sake ~]# mmmount BAGUETTE
Thu Oct 23 11:32:14 EDT 2014: mmmount: Mounting file systems ...
[root@sake ~]# mmlsmount BAGUETTE -L

File system BAGUETTE is mounted on 3 nodes:
  172.16.20.172   palinca
  172.16.20.170   sake
  172.16.20.171   olut
```

Now you can use the file system normally.

Note: This scenario intends to show a simple case of three nodes Spectrum Scale cluster. The implementation uses default settings for the cluster and file system creation. This scenario is not intended to be the standard way to set up a cluster. This scenario is just to show how to extend a Windows operating system-only cluster adding a Linux node.

3.6.18 Adding a Windows operating system node to the cluster

At this stage, our cluster is composed of three nodes. Two Windows NSD servers and one Linux NSD client. To complete the scenario, a Windows NSD client is added in this section to the Spectrum Scale cluster. When completed, there are four nodes in the cluster as shown in Figure 3-32 on page 149.

Note: At this point, the *sake* Windows server is added to a heterogeneous cluster. The steps explained in this section apply not only for a Windows operating system node being added to a cluster where Windows servers are the NSD but to any other cluster, regardless if the NSD servers are AIX or Linux.

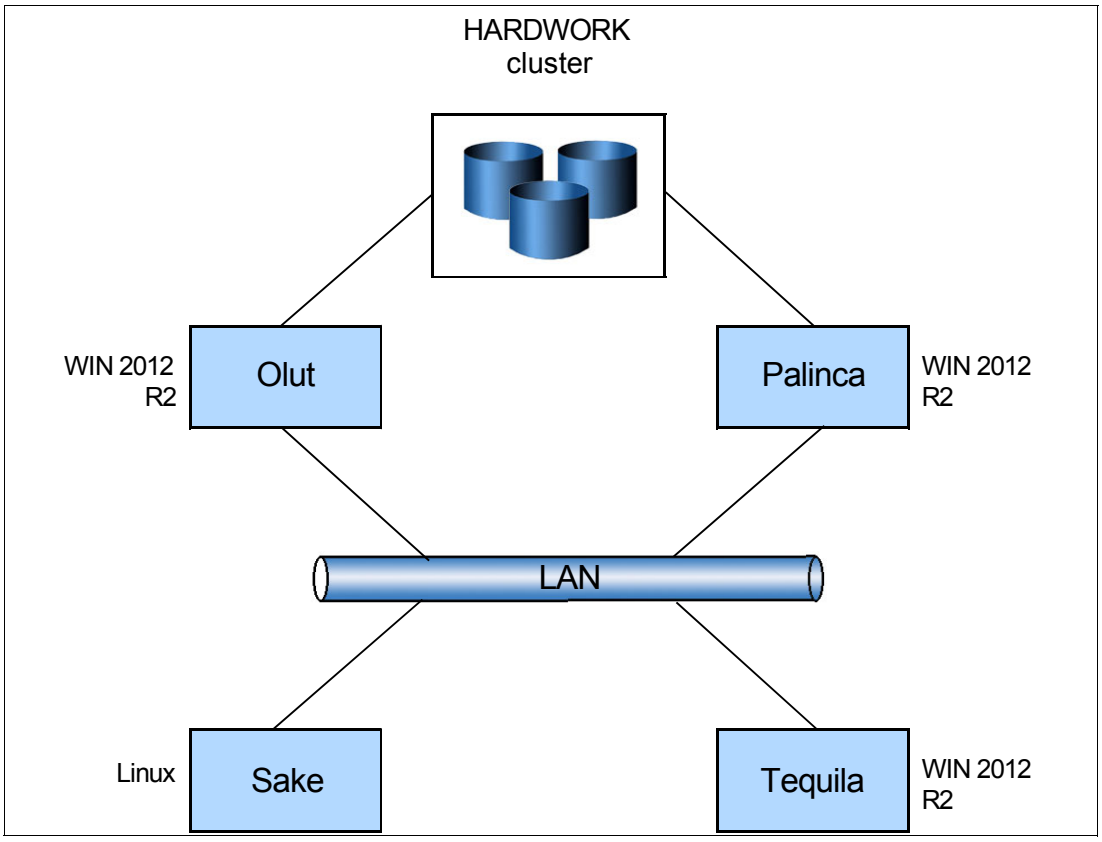


Figure 3-32 Windows servers and client plus Linux client cluster

There are several steps to installing IBM Spectrum Scale on Windows operating system nodes. In addition to fulfilling the software and hardware detailed in Chapter 2, “Infrastructure planning and considerations” on page 29, in the Windows operating system node, prepare the installation of IBM Spectrum Scale with the next steps:

- Configuring Windows 3.6.1, “Configuring Windows operating system” on page 113.
  - Static IP address 3.6.2, “Static IP address” on page 113.
  - Join node to the AD domain.
  - Account Control (UAC) 3.6.4, “UAC” on page 114.
  - Disable IPv6 3.6.5, “Disable IPv6” on page 115.
  - Windows firewall 3.6.6, “Windows operating system firewall” on page 118.
  - IBM Spectrum Scale traces auxiliary tools 3.6.7, “IBM Spectrum Scale traces auxiliary tools” on page 120.
- Installing Cygwin 3.6.8, “Installing Cygwin” on page 121.

When the above steps are completed, it is time to configure the IBM Spectrum Scale *mmwinserv* service. Use the **`mmwinservctl`** command from the node to be added to the cluster. See Example 3-47 on page 150.

Example 3-47 Configuring mmwinserv on the new Windows operating system node

```
Administrator@tequila:~ $ mmwinservctl set -N tequila --account HARDWORK/root --password "Redb00k;"
--remote-shell no

Node name        Service state  Remote shell  Account name
-----------------------------------------------------------
tequila          RUNNING        no            HARDWORK\root
Administrator@tequila:~ $ mmwinservctl query

Node name        Service state  Remote shell  Account name
-----------------------------------------------------------
tequila          RUNNING        no            HARDWORK\root
```

Because this is already a heterogeneous cluster where a non Windows operating system node exists, OpenSSH needs to be configured from the beginning. How to do so is explained in Example 3-37 on page 141. After following these instructions, we have a working OpenSSH server. By using the **scp** and the **cat** commands, is possible to add the public key of the new Windows operating system node to the `authorized_keys` file and distribute it to the rest of the nodes. See Example 3-48.

Example 3-48 Adding tequila SSH public key to the cluster and distributing it to all nodes

```
Administrator@tequila ~ $ cd .ssh

Administrator@tequila ~/.ssh $ scp root@sake:/root/.ssh/authorized_keys authorized_keys_3nodes
root@sake's password:
authorized_keys                                       100% 4083     4.0KB/s   00:00

root@tequila ~/.ssh $ cat id_rsa.pub >> authorized_keys_3nodes

root@tequila ~/.ssh $ cp authorized_keys_3nodes authorized_keys

root@tequila ~/.ssh $ scp authorized_keys root@olut:$PWD
The authenticity of host 'olut (172.16.20.171)' can't be established.
ECDSA key fingerprint is 09:b1:4b:9f:cd:58:65:c6:48:5b:0e:b4:13:c8:84:cd.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'olut,172.16.20.171' (ECDSA) to the list of known hosts.
root@olut's password:
authorized_keys                                       100% 4486     4.4KB/s   00:00

root@tequila ~/.ssh $ scp authorized_keys root@palinca:$PWD
The authenticity of host 'palinca (172.16.20.172)' can't be established.
ECDSA key fingerprint is 75:94:c8:24:fb:14:f1:90:74:0d:23:35:a0:35:1f:0a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'palinca,172.16.20.172' (ECDSA) to the list of known hosts.
root@palinca's password:
authorized_keys                                       100% 4486     4.4KB/s   00:00

root@tequila ~/.ssh$ scp authorized_keys root@sake:/root/.ssh/
root@sake's password:
authorized_keys                                       100% 4486     4.4KB/s   00:00
```

To check with **ssh** and **date** commands that the SSH communications and times work fine in all nodes, run the commands as shown in Example 3-49 on page 151.

Example 3-49 Checking ssh and time

```
$ ssh olut date; ssh palinca date; ssh sake date; ssh tequila date
Thu Oct 23 16:59:28 EDT 2014
Thu Oct 23 16:59:28 EDT 2014
Thu Oct 23 16:59:28 EDT 2014
Thu Oct 23 16:59:28 EDT 2014
```

Now it is possible to add the node to the cluster. This is done by running the **mmaddnode** command in any of the three nodes that belong to the Spectrum Scale cluster. When added, the client IBM Spectrum Scale license is needed, and accepted with the **mmchlicense** command. It is possible to see the nodes that belong to the cluster with the **mmlscluster** command as shown in Example 3-50.

Example 3-50 Adding the tequila node

```
# mmaddnode -N tequila
mmaddnode: Command successfully completed
mmaddnode: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmaddnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmchlicense client --accept -N tequila

The following nodes will be designated as possessing GPFS client licenses:
    tequila.hardwork.local
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         HARDWORK.hardwork.local
  GPFS cluster id:           10543779640873750826
  GPFS UID domain:           HARDWORK.hardwork.local
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name        IP address     Admin node name         Designation
--------------------------------------------------------------------------------
   1   olut.hardwork.local     172.16.20.171  olut.hardwork.local     quorum-manager
   2   palinca.hardwork.local  172.16.20.172  palinca.hardwork.local  quorum-manager
   3   sake.hardwork.local     172.16.20.170  sake.hardwork.local
   4   tequila.hardwork.local  172.16.20.174  tequila.hardwork.local
```

At this stage, use the **mmstartup** command to start IBM Spectrum Scale on the tequila node. Check when the node comes active with the **mmgetstate** command. Once the node is active, it is possible to mount the file system by using the **mmmount** command. And check if it is mounted with the **mmlsmount** command. See Example 3-51 on page 152.

Example 3-51 Starting IBM Spectrum Scale and mounting the file system in the tequila node

```
# mmstartup -N tequila
Thu Oct 23 17:05:50 EDT 2014: mmstartup: Starting GPFS ...
# mmgetstate -a -L -s

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      olut              1        2          4       active      quorum node
       2      palinca           1        2          4       active      quorum node
       3      sake              1        2          4       active
       4      tequila           1        2          4       active

 Summary information
---------------------
Number of nodes defined in the cluster:            4
Number of local nodes active in the cluster:       4
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     2
Number of quorum nodes active in the cluster:      2
Quorum = 1*, Quorum achieved

# mmmount BAGUETTE -N tequila
Thu Oct 23 17:07:45 EDT 2014: mmmount: Mounting file systems ...

# mmlsmount BAGUETTE -L

File system BAGUETTE is mounted on 4 nodes:
  172.16.20.172   palinca
  172.16.20.170   sake
  172.16.20.171   olut
  172.16.20.174   tequila
```

At this point, it is possible to use the file system normally in all nodes that have it mounted.

3.7 Oracle Real Application Cluster with IBM Spectrum Scale

IBM Spectrum Scale enhanced scalability, availability, and performance features provide a robust concurrent file system solution for clustered applications requiring active-active processing, which scales up horizontally, such as Oracle Real Application Clusters (RACs).

Advantages of running Oracle RAC on IBM Spectrum Scale file systems are as follows:

- Simplified installation and configuration.
- Possibility of using the AUTOEXTEND option for Oracle data files, similar to JFS/JFS2 installation.
- Ease of monitoring free space for data files storage.
- Ease of running cold backups and restores, similar to a traditional file system.
- Capability to place Oracle binary files on a shared file system, with several advantages for the patching process.
- Considering the direct I/O feature, performance for running Oracle RAC on Spectrum Scale is almost the same as performance of Oracle using raw devices.

- IBM Spectrum Scale support for Persistent Reserve improves failover functionality and allows for the Oracle OCR and Voting disks to be placed on the IBM Spectrum Scale file system.
- IBM Spectrum Scale introduces unparallel ILM capabilities to the file system.

IBM Spectrum Scale can be used for storing all Oracle RAC file types (for database and clusterware products):

- Oracle clusterware:
  - Clusterware binary files
  - Clusterware registry files
  - Clusterware voting files
- Oracle Database:
  - Database binary files
  - Database initialization files (`init.ora` or `spfile`)
  - Control files
  - Data files
  - Redo log files
  - Archived redo log files
  - Flash recovery area files

Oracle RAC with IBM Spectrum Scale can be deployed on AIX partitions, or on Linux on Power partitions (only for Oracle RAC 10gR2). When planning for Oracle RAC with IBM Spectrum Scale, see the latest IBM Spectrum Scale updates and recommendations on IBM Spectrum Scale FAQs:

`http://ibm.co/1IKO6PN`

A sample Oracle RAC on an AIX environment that uses IBM Spectrum Scale is illustrated in Figure 3-33 on page 154.

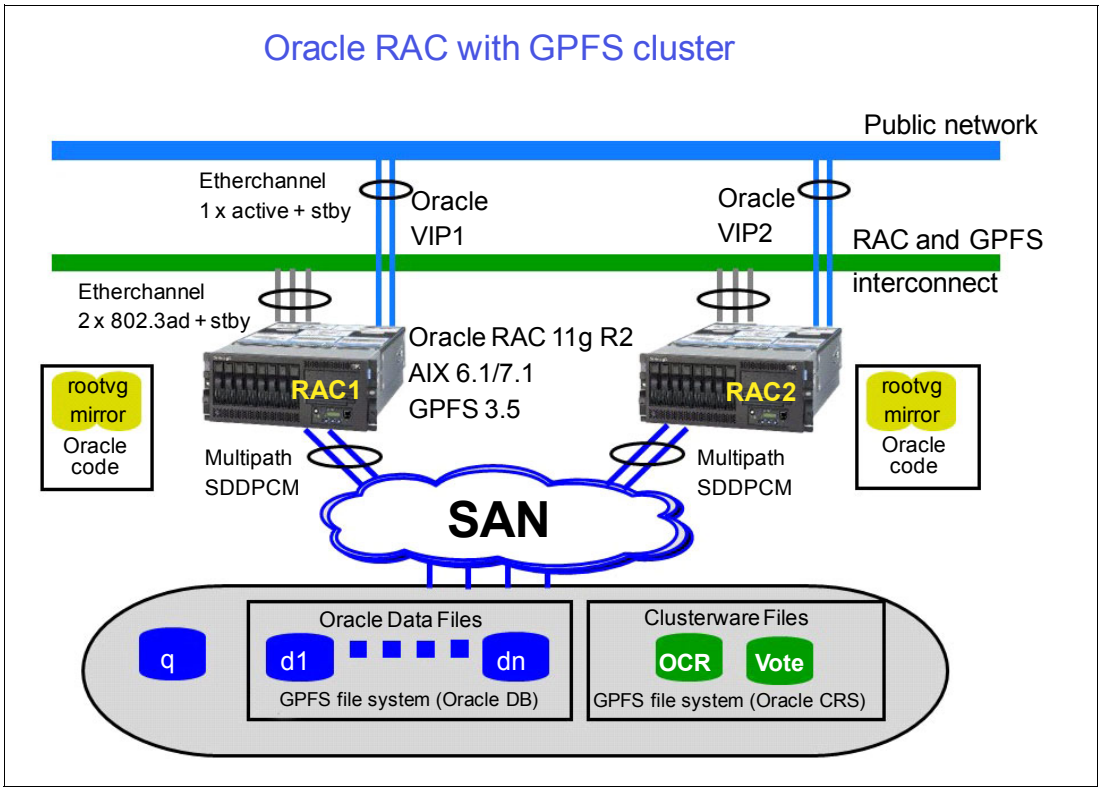


Figure 3-33 Oracle RAC cluster using IBM Spectrum Scale

The following characteristics apply for the Oracle RAC environment on AIX with IBM Spectrum Scale:

- IBM Spectrum Scale can use only NSDs with direct storage attachment for the Oracle database and clusterware files. Other topologies, such as using NSD client/server access to the shared file system, are not supported.
- It is generally recommended to use distinct networks for IBM Spectrum Scale and Oracle RAC interconnect. However, due to the limited traffic of Spectrum Scale using all nodes that are directly attached to the storage, Oracle RAC and IBM Spectrum Scale can share the network. An important consideration for this network is the availability, as Oracle RAC (before 11.2.0.2) and IBM Spectrum Scale rely exclusively on the OS/hardware availability mechanisms. We recommend using link aggregation (LACP/802.3ad) interfaces with Ethernet adapters (GbE and 10 GbE are supported) or using InfiniBand for larger deployments requiring a high performance and low latency network. Also, see Oracle RAC Technologies Certification Matrix for UNIX Platforms at the following site:

  `http://bit.ly/1NGOJLO`

  Figure 3-34 on page 161 shows an example of GbE network for Spectrum Scale and RAC interconnect, using three interfaces on each node. Two of the interfaces are grouped in an LACP (802.3ad), requiring to be connected to the same switch and a third interface as backup, which is connected to a different switch. This basic configuration provides protection for both interface and switch failures. In version 11.2.0.2, Oracle introduces the Oracle Redundant Interconnect feature, providing an alternative for high availability of the Oracle interconnect network.
- Public network is used by Oracle RAC with Virtual IP addresses (VIP1 and VIP2 in Figure 3-34 on page 161). You can use Network Interface Backup interfaces with Ethernet adapters for protecting the node access to the public network in case of port/switch failures. Besides the node VIPs addresses on the public network, Oracle RAC 11gR2

introduces SCAN VIPs, which map multiple failover virtual IPs to a single name used for client connection to the Oracle RAC database.

- Oracle clusterware files must reside on shared storage. There are several options for the Oracle clusterware files (OCR and voting files):
  - Use raw devices (AIX hdisk)
  - Use the Spectrum Scale file system. We recommend using a distinct file system for Oracle clusterware files from the regular database files. You can use multiple file systems on distinct storage partitions to distribute the OCR and vote files. 11gR2 supports up to five copies of the OCR file and multiple vote disks. We strongly recommend using Persistent Reservation with Spectrum Scale for the NSDs when storing the clusterware files on Spectrum Scale.
  - Use Oracle Automatic Storage Management (ASM). It requires 11gR2, or later. See more details in the Oracle Help Center:

    https://docs.oracle.com/cd/E11882_01/rac.112/e41959/votocr.htm#CWADD92368

Note: Starting with Oracle RAC 11gR2, Oracle Universal Installer does not support installing the Oracle clusterware files on raw devices, but they are supported when migrating from previous versions. With release 12cR1, Oracle no longer supports the use of raw devices with the database and clusterware (see My Oracle Support Doc ID 578455.1).

- Oracle database files must be shared between the cluster nodes. They use the IBM Spectrum Scale file system for concurrent access of the database RAC instances running on them. These files include: data files, control files, and redolog files. The archive logs and the Flash Recovery Area should also be placed on IBM Spectrum Scale to enable access of all cluster nodes in case of recovery.
- Oracle clusterware and database binary files can be installed on the local file system (jfs2) or IBM Spectrum Scale. When installed on Spectrum Scale, the Oracle patching process applies once for all the nodes. However, in case of placing the binary files on the local file system on each node, rolling migration/patching is an available option. Oracle recommends that the clusterware home directory (binary files) be placed on local file systems. See My Oracle Support Doc ID 1376369.1.

Oracle RAC and IBM Spectrum Scale supported versions

Table 3-1 shows the combinations of versions for Oracle RAC, AIX, and IBM Spectrum Scale, which can be used for Oracle RAC deployments with IBM Spectrum Scale.

Note: IBM currently supports only IBM Spectrum Scale 3.5 and 4.1, and AIX 6.1 and 7.1

Table 3-1 Certified combinations of Oracle RAC, AIX, and IBM Spectrum Scale

Oracle RAC	IBM Spectrum Scale	AIX
10gR2	GPFS 3.1/GPFS 2.3	AIX 5.3
	GPFS 3.2	AIX 5.3, AIX 6.1
11gR1	GPFS 3.1	AIX 5.3
	GPFS 3.3/GPFS 3.2	AIX 5.3, AIX 6.1

Oracle RAC	IBM Spectrum Scale	AIX
11gR2	GPFS 3.2	AIX 5.3, AIX 6.1
	GPFS 3.3/3.4	AIX 5.3, AIX 6.1, AIX 7.1
	GPFS 3.5	AIX 6.1, AIX 7.1
	Spectrum Scale 4.1 (see Note)	AIX 6.1, AIX 7.1
12cR1	GPFS 3.4	AIX 6.1, AIX 7.1
	GPFS 3.5	AIX 6.1, AIX 7.1

Note: At the time of writing this IBM Redbooks publication, IBM Spectrum Scale 4.1 is certified for use with Oracle RAC 11gR2 (11.2.0.4). For more updates on the certified Oracle and IBM Spectrum Scale versions, search My Oracle Support certification listings for IBM Spectrum Scale. Also, review note 1376369.1 for more information about minimum maintenance levels and patches required.

Oracle RAC and PowerVM virtualization support

Oracle RAC can run on IBM Power Systems using virtualization technologies such as: LPARs, IBM Micro-Partitioning®, dynamic LPAR, and VIOS. These features are part of the PowerVM offering. IBM Spectrum Scale supports the PowerVM features. See more details in 1.7, “IBM Spectrum Scale and virtualization” on page 18 and also refer to the IBM Spectrum Scale FAQs document for more updates at this website:

http://ibm.co/1IKO6PN

The following virtualization technologies are available for use with Oracle RAC (restrictions can apply for some versions as indicated):

- Logical partitions (LPARs): These are subsets of computer’s processors, memory, and I/O resources that can be operated independently with its own operating system and applications. LPARs can use:
  - Dedicated processor partitions: LPARs use dedicated processors that are assigned to a single LPAR.
  - Shared processor partitions: LPARs use Micro-Partitioning with a shared processor pool. Micro-Partitioning divides a physical processor’s computing power into fractions of a processing unit and shares them among logical partitions. Processing capacity can be configured in fractions of 1/100 of a processor. The minimum amount of processing capacity that has to be assigned to a partition is 1/10 of a processor, or 1/20 with IBM POWER7+™ and POWER8. A shared processor pool is a group of physical processors that are sharing the processing power to multiple LPARs.
- Dynamic LPARs (DLPARs): Allows a shared or dedicated LPAR to dynamically change the number of processors, memory, and virtual or physical adapters without rebooting the physical server.
  - For dedicated processor partitions, you can dynamically add, move, or remove whole processors.
  - For shared processor partition, you can also dynamically change the shared processor capacity, the weight of the uncapped attribute, virtual processors, and capped and uncapped mode.

- Virtual I/O Server: It allows sharing of physical resources between LPARs including virtual SCSI and virtual networking. Considerations for using VIOS with Oracle RAC:
  - Use dual-VIOS configuration on each physical server for high availability.
  - Virtual Ethernet can be used for all cluster networks: IBM Spectrum Scale, Oracle clusterware public and private interconnects.
  - Shared Ethernet Adapters (SEAs) can be used to bridge the VIOS client LPARs to the external network resources. In a dual-VIOS configuration, you can use SEA failover for protecting the client access to the external networks in case of VIOS failure. The following switch settings are suggested for the ports where SEA adapters are connected. They enable a faster failover of network interfaces and avoid Oracle cluster node evictions due to the topology services (CSS) daemon network timeout when recovering from failures:
    - Spanning tree protocol = enable
    - Start port fast = enable
    - Delay forwarding = 15s
    - Trunking = off
    - Etherchannel = off
  - Disk attachment that uses NPIV or Virtual SCSI can be used in supported combinations of VIOS, IBM Spectrum Scale, and Oracle RAC versions. For more information, consult “IBM General Parallel File System (GPFS) and Oracle RAC (Doc ID 1376369.1)” on My Oracle Support. An advantage when using NPIV is that you can use persistent reservation, which is highly recommended for Oracle RAC with IBM Spectrum Scale.
- Live Partition Migration (LPM): It allows you to migrate running AIX and Linux LPARs and their hosted applications from one physical server to another without disrupting the infrastructure services. The migration transfers the entire partition state, including the processor context, memory, attached virtual devices, and connected users. Currently, LPM is certified with Oracle RAC 11gR2 on AIX using IBM Spectrum Scale or ASM.
- IBM Active Memory™ Expansion (AME): This is a new feature available on POWER7 and POWER8 systems, which provides a transparent mechanism for extending the memory of a physical server above the real capacity by using memory compression technology. It is currently supported by Oracle RAC for version 11gR2.
- Workload Partition (WPAR): It provides the capability to run multiple isolated application environments within the same host operating system in a logical partition. This feature is currently being evaluated for support with Oracle RAC.

Note: For more information about Oracle supported virtualization technologies, see *Supported Virtualization and Partitioning Technologies for Oracle Database and RAC Product Releases (UNIX and Linux Operating Systems)*:

`http://www.oracle.com/technetwork/database/virtualizationmatrix-172995.html`

For more information about IBM PowerVM technology, see the publication *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940, available on:

`http://www.redbooks.ibm.com/abstracts/sg247940.html`

Preferred practices for IBM Spectrum Scale with Oracle RAC

Optimizing the Spectrum Scale configuration for a particular application or database is specific to the I/O workload type of that application or database. With Oracle RAC configurations using Spectrum Scale, several recommendations and preferred practices apply:

- Using LUNs on RAID devices: a single LUN should be configured for each array. Do not stripe a LUN over multiple arrays because Spectrum Scale has its own striping mechanism.
- Use the Persistent Reservation, if supported by the storage subsystem and host attachment. It brings significant improvements for the recovery times during various failover cases, such as a node down event. This feature is particularly important when placing Oracle clusterware files on IBM Spectrum Scale.
- Use the following Spectrum Scale block sizes:
  - 512 KB is generally suggested.
  - 256 KB can be used if there is significant file activity other than Oracle, or there are many small files not part of the database.
  - 1 MB recommended for file systems larger than 100 TB. See My Oracle Support Doc ID: 302806.1.

  Notes:

  - Do not set the Spectrum Scale block size to the Oracle "db_block_size".
  - Set the Oracle "db_block_size" equal to the LUN segment size or a multiple of the LUN pdisk segment size.
  - Set the Oracle **`init.ora`** parameter "db_file_multiblock_read_count" value to pre-fetch one or two full Spectrum Scale blocks. For example, if your Spectrum Scale block size is 512 KB and the database block size is 16 K, set the value of this parameter to either 32 or 64 db blocks.

- Spectrum Scale worker threads allow the maximum parallelism of the Oracle AIO threads:
  - prefetchThreads is for large sequential file I/O, whereas worker1Threads is for random, small file I/O.
  - In GPFS 3.3 and later: prefetchThreads+worker1Threads+nsdMaxWorkerThreads < 1500, where nsdMaxWorkerThreads is the maximum number of NSD threads on an NSD server that is concurrently transferring data with NSD clients. Tuning this parameter is not relevant with Oracle RAC environments where NSDs use only direct storage access, but it can influence on the total number of threads of the three classes. Following are the default values of the parameters:
    - prefetchThreads=72
    - worker1Threads=48
    - nsdMaxWorkerThreads=64 (GPFS 3.4) or 512 (GPFS 3.5/4.1).

    The default value of worker1Threads is usually not sufficient in Oracle RAC environments, and it should be increased. You can use the following commands to determine how many prefetchThreads and worker1Threads are in use and how many application requests are waiting:

    For the prefetchThreads: **`mmfsadm dump fs | egrep "nPrefetchThreads:|total wait"`**

    For the worker1Threads: **`mmfsadm dump mb | grep Worker1Threads`**

- As a starting point, when requiring Spectrum Scale sequential I/O, set the prefetchThreads 50 - 100 (the default is 72), or let it on the default value. Set the worker1Threads to the difference of (548 - prefetchThreads).

- The number of AIX AIO threads to create is approximately the same as the Spectrum Scale worker1Threads setting:
  - The AIX AIO maxservers setting is the number of kprocs per CPU. The suggested value of this parameter is worker1Threads divided by the number of CPUs in the LPAR.

    On AIX 6.1, and later you can change dynamically the **maxserver** parameter using the **ioo** command: **ioo -p -o aio_maxservers=<value>**. In AIX 6.1 server threads are no longer allocated statically, AIO servers are started and stay active as long as they service I/O requests.

- Oracle RAC and Direct I/O on IBM Spectrum Scale. By default, Oracle uses the Asynchronous I/O (AIO) and Direct I/O (DIO) features of AIX to do its own I/O scheduling directly to disks, bypassing Spectrum Scale caching and prefetching facilities. Therefore:
  - Do not use the “dio” mount option for the Spectrum Scale file system or change the DIO attribute for any Oracle files.
  - The Oracle **init.ora** parameter “filesystemio_options” setting is ignored for Oracle files on Spectrum Scale.
  - DISK_ASYNC_IO should always be set to TRUE (default value).

- IBM Spectrum Scale and pinned SGA. Oracle databases requiring high performance usually benefit from running with a pinned Oracle SGA. This is also true when running with IBM Spectrum Scale since IBM Spectrum Scale uses DIO, which requires that the user I/O buffers (in the SGA) be pinned. IBM Spectrum Scale would normally pin the I/O buffers on behalf of the application but, if Oracle has already pinned the SGA, Spectrum Scale recognizes this and does not duplicate the pinning, saving additional system resources. Pinning the SGA on AIX requires the following three steps:

  a. `/usr/sbin/vmo -r -o v_pinshm=1`

  b. `/usr/sbin/vmo -r -o maxpin%=percent_of_real_memory`

     Where percent_of_real_memory = ((size of SGA / size of physical memory) *100) + 3

  c. Set the **LOCK_SGA** parameter to TRUE in the init.ora.

- Other important Spectrum Scale file system attributes:
  - If the Spectrum Scale file system contains a shared Oracle database home or clusterware home, the default value for the maximum number of inodes might be insufficient for the Oracle Universal Installer (OUI) installation process. Use a larger value for the inode number, typically 50000.
  - For Oracle RAC node recovery to work correctly, Spectrum Scale must be configured to be automatically loaded at boot time (**mmchconfig autoload=yes**) and automatically mount the Spectrum Scale file systems (**mmchfs <gpfsdev> -A yes**).

3.8 IBM Spectrum Scale integration with IBM Spectrum Protect (formerly Tivoli Storage Manager)

IBM Spectrum Scale is used together with IBM Spectrum Protect software to provide an enhanced backup solution, leveraging the capabilities of both products to meet the requirements of the enterprises environments.

Following are the main integration features between the two products:

- Backup of Spectrum Scale file systems using the Spectrum Protect Backup-Archive client. The Spectrum Protect client recognizes the Spectrum Scale file system when performing a backup operation and the Spectrum Protect server associates the file system to a file space of type *mmfs*. Spectrum Scale attributes of files and directories, such as storage pool, fileset associations, data, and metadata replicas are automatically saved by the Spectrum Protect client and retrieved during a restore operation (assuming the Spectrum Scale file system configuration is the same). See the **mmlsattr** command for further details about Spectrum Scale attributes. Spectrum Protect client also supports Spectrum Scale ACLs depending on the platform. See more details about ACL support and restrictions for the latest Spectrum Protect client version 7.1, on the following site:

  http://ibm.co/1Gae9By

  A particular case for backing up Spectrum Scale data is including the Spectrum Protect server in the Spectrum Scale cluster. In this case, Spectrum Protect server has access to the Spectrum Scale file system and can act as a data mover node, backing up and restoring data to and from the backup devices.

  You can enhance the backup operation of Spectrum Scale file systems using the **mmbackup** tool, which is based on the Spectrum Protect client. In this case, Spectrum Scale provides unique enhancements for the Spectrum Protect backups comparing with the backup of the regular file systems:

  – IBM Spectrum Scale uses the policy engine to detect the changed, new, and deleted files, which are written to multiple lists.

  – Lists of changed, new, and deleted files are passed to Spectrum Protect for processing.

  See more details about the **mmbackup** command, backup, and restore scenarios in 7.3, “Backup and restore for IBM Spectrum Scale” on page 373.

- Implementing IBM Spectrum Scale ILM with external storage pool together with Spectrum Protect for Space Management (also known as *Spectrum Protect HSM*). Currently, it is supported for AIX and Linux x86_64 platforms. See further details about HSM and Spectrum Scale supported versions on the following site:

  http://www-01.ibm.com/support/docview.wss?uid=swg21321200

  You can set up external storage pools and Spectrum Scale policies allowing the Spectrum Scale policy manager to coordinate the file migrations from a native Spectrum Scale online pool to external pools on the Tivoli Storage Manager server. The Spectrum Scale policy manager invokes the migration through the HSM client command-line interface. The migration candidate selection is identical to the Spectrum Scale native pool to pool migration rule. The Policy Engine uses scripts to call the Spectrum Protect command **dsmmigrate** for the migration of files from a native storage pool to the Spectrum Protect server. Two different approaches can be used to drive an HSM migration via Spectrum Scale policies. The two approaches are only different in how the **mmapplypolicy** command, which does the policy scan, is started:

  – Manual Spectrum Scale driven migration: The manual Spectrum Scale driven migration is performed when the user or a UNIX cron job executes the **mmapplypolicy** command with a predefined migration policy.

  – Automatic Spectrum Scale threshold migration: The Spectrum Scale threshold migration is performed when the user has specified a threshold policy and the Spectrum Scale policy daemon is enabled to monitor the storage pools in the file system for that given threshold. If a predefined high threshold is reached, which means the filling level of the storage pool reached the predefined high water mark, the monitor daemon automatically starts the **mmapplypolicy** command to perform an inode scan.

The following additional features are provided with Spectrum Scale external pools by using Spectrum Protect HSM:

- Migrate and recall operations can be processes in parallel using multiple Spectrum Scale nodes.
- Transfer of file during migrate/recall operations can be done using the SAN network by the Tivoli Storage Agent (LAN-free transfer).

For more information about Spectrum Scale ILM, external pools and policies, refer to Chapter 5, “Information lifecycle management” on page 245.

- IBM Spectrum Scale can be used as disk storage pool in the Spectrum Protect server hierarchy. One useful case is using LAN-free transfers of client backup data to a disk storage pool in Spectrum Protect residing on a Spectrum Scale file system. The following section provides more details about implementing a Spectrum Protect LAN-free backup configuration using disk pools shared with Spectrum Scale.

Spectrum Protect LAN-free backup using IBM Spectrum Scale

In this scenario, IBM Spectrum Scale provides file sharing over the SAN for the Spectrum Protect disk pools, thus enabling the LAN-free transfer of data between the Spectrum Protect clients and server. Figure 3-34 shows the control and data paths of the backup traffic from the client to the disk storage pool. Observe that Spectrum Protect is using the *FILE* device class for the disk pool and Spectrum Scale shares the disk access to the LAN-free clients. Clients are writing data in files managed by Spectrum Protect, which reside in the Spectrum Scale file system.

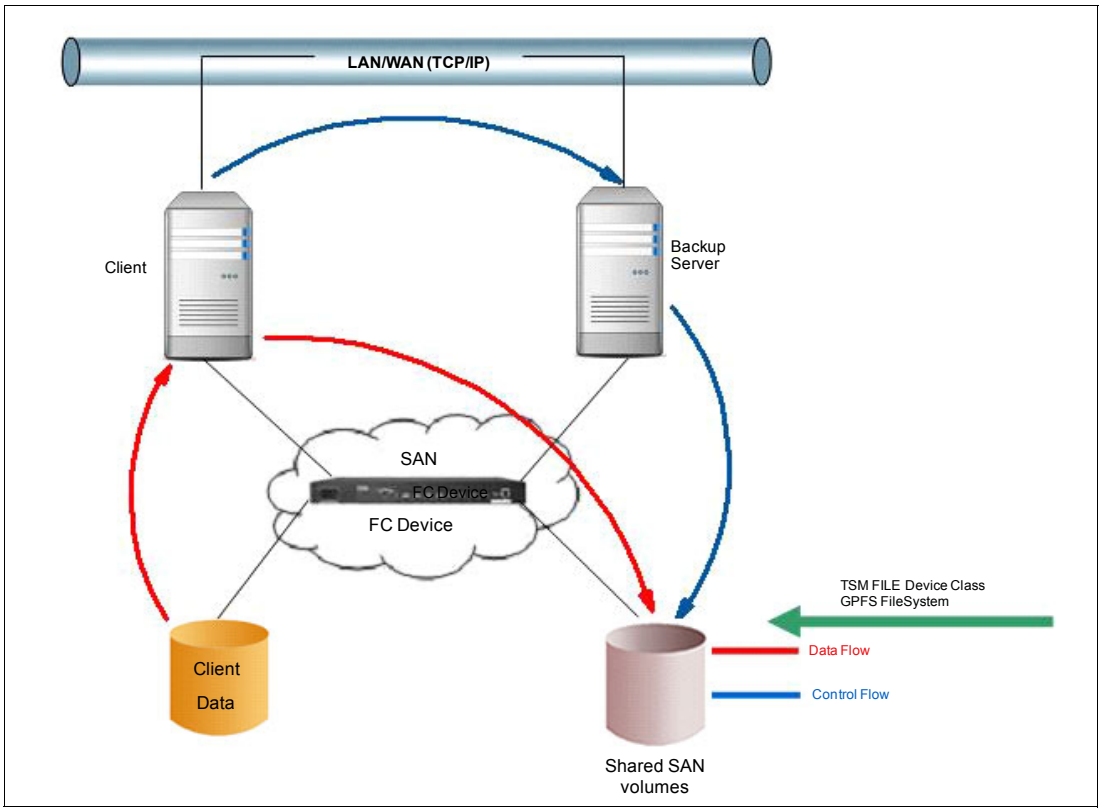


Figure 3-34 Spectrum Protect LAN-free with Spectrum Scale

Implementing Spectrum Protect LAN-free with Spectrum Scale

Before implementing Spectrum Scale in a Spectrum Protect LAN-free environment, consider the following aspects:

- All nodes (servers) performing LAN-free transfer are part of the same Spectrum Scale cluster with the Spectrum Protect server and share all the volumes of the Spectrum Protect disk pools targeted for LAN-free backup.
- If the clients are already part of a Spectrum Scale cluster, you should consider adding the Spectrum Protect server in the cluster, if supported by your environment. This use case would not require the Spectrum Protect LAN-free backup, as long as Spectrum Protect server mounts the client Spectrum Scale file systems, and performs the file level backup to a Spectrum Protect storage pool (disk, tape).
- IBM Spectrum Scale does not support heterogeneous scenarios for direct storage attachment. While the architecture of Spectrum Scale generally allows sharing of LUNs between different operating systems (Linux, AIX, and Windows operating systems based on Spectrum Scale NSD), the actual implementation of various OS-specific features prevents this from being used at the current time. See the concurrent access for SAN-attached disk considerations in IBM Spectrum Scale FAQ:

  http://ibm.co/1IKO6PN
- You need to assign at least one quorum node in the Spectrum Scale cluster. We recommend defining the Spectrum Protect server as a quorum node. The availability of the Spectrum Scale cluster needs to be correlated with the availability of the Spectrum Protect server. Client data backup/restore operations are not possible if the Spectrum Protect services are not available. For Spectrum Scale availability reasons, use two quorum nodes and enable the use of the tiebreaker disk. The assignment of the quorum nodes is also related to the IBM Spectrum Scale licenses. All quorum nodes must use the IBM Spectrum Scale server license. The rest of the nodes in this case, not having a manager role in the IBM Spectrum Scale cluster can use the IBM Spectrum Scale client license.
- The operating system requirements must meet both IBM Spectrum Protect and IBM Spectrum Scale requirements. LAN-free using IBM Spectrum Scale is supported by Tivoli Storage Manager 6.1 or later versions. See IBM Spectrum Scale requirements for each platform in Chapter 2, “IBM Spectrum Scale supported platforms” on page 56. See Spectrum Protect requirements for LAN-free at the following site:

  http://www-01.ibm.com/support/docview.wss?&uid=swg21243309

Example scenario

The following scenario provides practical guidance for implementing Spectrum Protect LAN-free with IBM Spectrum Scale. The preliminary steps of installing the Spectrum Protect software as well as IBM Spectrum Scale are assumed as covered. We provide an example for the relevant configurations in Spectrum Scale and Spectrum Protect.

The scenario is illustrated in Figure 3-35 on page 163 and it consists of the following components:

- Spectrum Protect server node: *tsmsrv* with the following software installed: Spectrum Protect Extended Edition V7.1, AIX 7.1, Spectrum Scale 4.1.
- Spectrum Protect clients: *tsmcl1*, *tsmcl2*, and *tsmcl3* with the following software installed: Spectrum Protect Storage Agent 7.1, AIX 7.1, Spectrum Scale 4.1.
- Shared storage connected to Spectrum Protect server and clients: four 100-GB LUNs. One of the LUNs will also be used as Spectrum Scale cluster quorum tiebreaker disk.

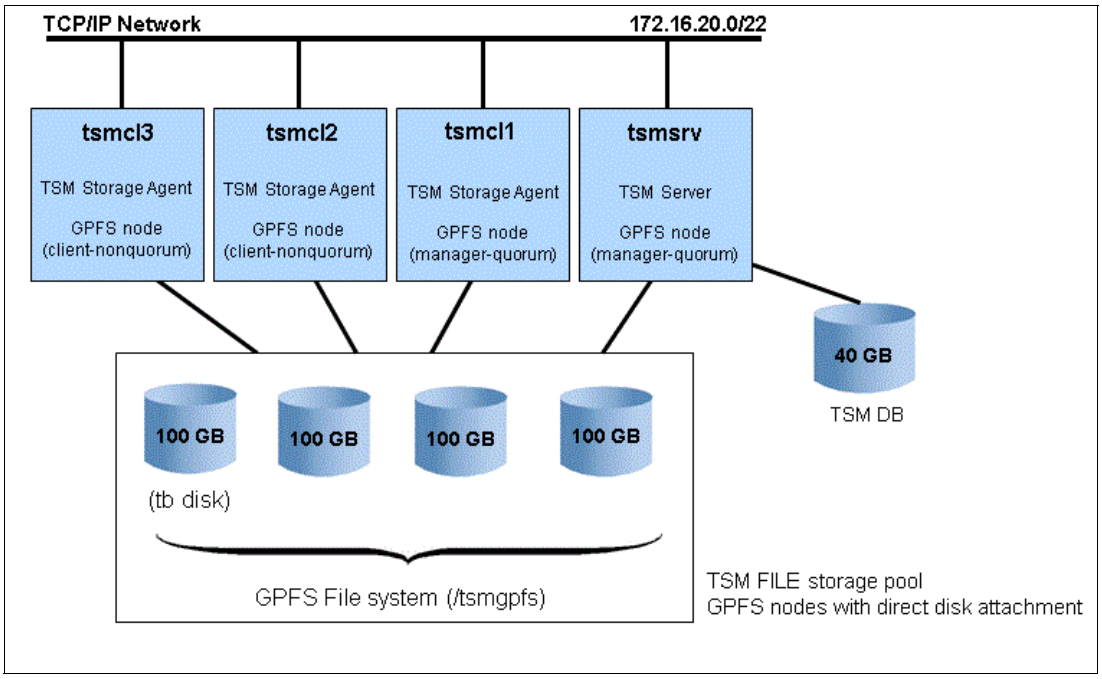


Figure 3-35 Spectrum Protect LAN-free with Spectrum Scale scenario

The following sections present the configurations that are performed on Spectrum Scale and Spectrum Protect.

The following steps are performed to configure Spectrum Scale:

1. We create the Spectrum Scale cluster containing the following nodes: tsmsrv, tsmcl1, tsmcl2, and tsmcl3. In our scenario, we use one quorum node associated with the Spectrum Protect server node and the second one with *tsmcl1*. The input file of nodes contains Spectrum Protect server and client nodes as follows:

   ```
   tsmsrv:manager-quorum:
   tsmcl1:manager-quorum:
   tsmcl2:client-nonquorum:
   tsmcl3:client-nonquorum:
   ```

   Example 3-52 shows the output of the **mmcrcluster** command that is used for creating the Spectrum Scale cluster.

Example 3-52 Creating the Spectrum Scale cluster

```
root@tsmsrv:/> mmcrcluster -N /gpfs/nodes.aix --ccr-enable -r /usr/bin/ssh -R
/usr/bin/scp -C tsmgpfs -A
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Processing the rest of the nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
             Use the mmchlicense command to designate licenses as needed.
mmcrcluster: Propagating the cluster configuration data to all
             affected nodes.  This is an asynchronous process.
```

2. We apply the IBM Spectrum Scale licenses for the node roles. See Example 3-53.

Example 3-53 Applying the IBM Spectrum Scale licenses to the cluster nodes

```
root@tsmsrv:/>mmchlicense server --accept -N tsmsrv,tsmcl1
The following nodes will be designated as possessing GPFS server licenses:
        tsmsrv
        tsmcl1
mmchlicense: Command successfully completed
mmchlicense: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

root@tsmsrv:/>mmchlicense client --accept -N tsmcl2,tsmcl3
The following nodes will be designated as possessing GPFS client licenses:
        tsmcl2
        tsmcl3
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. We create the NSDs consisting of the volumes used for Spectrum Protect disk pool. The input file is shown in Example 3-54.

Example 3-54 NSD creation input file

```
root@tsmsrv:/gpfs>cat gpfs.dsk
%nsd:device=hdisk3 usage=dataAndMetadata nsd=tsmdisk1
%nsd:device=hdisk4 usage=dataAndMetadata nsd=tsmdisk2
%nsd:device=hdisk5 usage=dataAndMetadata nsd=tsmdisk3
%nsd:device=hdisk6 usage=dataAndMetadata nsd=tsmdisk4
```

The output of the **mmcrnsd** command for defining the NSDs in shown in Example 3-55.

Example 3-55 Creating the NSDs

```
root@tsmsrv:/gpfs>mmcrnsd -F ./gpfs.dsk
mmcrnsd: Processing disk hdisk3
mmcrnsd: Processing disk hdisk4
mmcrnsd: Processing disk hdisk5
mmcrnsd: Processing disk hdisk6
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

4. We change the cluster configuration to use tiebreaker disks. We use in our example one of the disks designated for file system use. See Example 3-56.

Example 3-56 Change cluster quorum for using tiebreaker disks

```
root@tsmsrv:/gpfs>mmchconfig tiebreakerdisks=tsmdisk1
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

5. We start the cluster services using the **mmstartup -a** command and create the Spectrum Scale file system for LAN-free transfers. In Example 3-57, we create a file system with 1 MB block size.

Example 3-57 Creating the Spectrum Scale file system

```
root@tsmsrv:/gpfs>mmstartup -a
Fri Oct 17 16:27:52 EDT 2014: mmstartup: Starting GPFS ...
root@tsmsrv:/gpfs>mmcrfs tsmgpfs -F /gpfs/gpfs.dsk -A yes -B 1M -T /tsmgpfs

The following disks of tsmgpfs will be formatted on node tsmsrv:
    tsmdisk1: size 102400 MB
    tsmdisk2: size 102400 MB
    tsmdisk3: size 102400 MB
    tsmdisk4: size 102400 MB
Formatting file system ...
Disks up to size 923 GB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/tsmgpfs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. We mount the file system on all Spectrum Scale nodes using: **mmmount tsmgpfs -a**.

The following steps are performed for Spectrum Protect configuration:

1. We change the ownership of the Spectrum Scale file system to the Spectrum Protect user instance, on the Spectrum Protect server node (in our case, tsminst1 with primary group tsmsrvs):

   ```
   chown tsminst1:tsmsrvs /tsmgpfs
   ```

   Note: The directory on the Spectrum Scale file system used as Spectrum Protect storage pool for LAN-free operations must be owned by the Spectrum Protect user instance. This operation is performed on the Spectrum Protect server node. The user does not need to be defined on the other Spectrum Scale cluster nodes. For security reasons, we recommend choosing the user and group IDs so that they do not interfere with the existing uid/gid on the client systems.

2. We define a FILE device class in Spectrum Protect associated with the Spectrum Scale file system `/tsmgpfs`:

   ```
   define devclass gpfsdevcl devtype=FILE mountlimit=64 maxcap=10G
   directory=/tsmgpfs shared=yes
   ```

   When creating the devclass with option *shared=yes*, it automatically creates a FILE library with the same name (in our example GPFSDEVCL) and the number of drives specified by the **mountlimit** parameter (in our example: GPFSDEVCL1 ... GPFSDEVCL64).

3. We define the storage pool for the LAN-free backup. When using the **maxscratch** parameter in the definition of the storage pool, you need to correlate it with *maxcap* value from the previous step, so *maxcap x maxscratch* should not exceed the total capacity of the Spectrum Scale file system.

   ```
   define stg gpfspool GPFSDEVCL maxscratch=40
   ```

4. We change the destination pool in the Spectrum Protect policy to the defined LAN-free storage pool. In our case, we have all Spectrum Protect clients registered in policy domain GPFS_DOM, and we set the copygroup of the *standard* management class with destination pool=GPFSPOOL.

   ```
   update copy GPFS_DOM standard standard standard destination=GPFSPOOL
   ```

   For activating the policy changes, run the following command:

   ```
   activate policy GPFS_DOM standard
   ```

LAN-free enablement

This section describes the steps to enable the LAN-free configuration.

1. On the Spectrum Protect server, we set the server password and define the storage agents as Spectrum Protect servers. See Example 3-58.

Example 3-58 Defining the storage agents in Spectrum Protect

```
tsm: TSMGPFS1>set serverpa XXXXXX
ANR2131I Server password set.

tsm: TSMGPFS1>define server TSMCL1_STA hla=172.16.20.154 lla=1500 serverpa=XXXXX
ANR1660I Server TSMCL1_STA defined successfully.

tsm: TSMGPFS1>define server TSMCL2_STA hla=172.16.20.155 lla=1500 serverpa=XXXX
ANR1660I Server TSMCL2_STA defined successfully.

tsm: TSMGPFS1>define server TSMCL3_STA hla=172.16.20.156 lla=1500 serverpa=XXXX
ANR1660I Server TSMCL3_STA defined successfully.
```

2. Define paths from each storage agent to all drives in the FILE library. Example 3-59 provides an output for defining a single path between the storage agent TSMCL1_STA and the drive GPFSDEVCL1 in the FILE library GPFSDEVCL.

Example 3-59 Define a path for storage agent to the Spectrum Scale file system

```
tsm: TSMGPFS1>define path TSMCL1_STA GPFSDEVCL1 srct=server destt=drive
library=GPFSDEVCL device=FILE directory=/tsmgpfs
ANR1720I A path from TSMCL1_STA to GPFSDEVCL GPFSDEVCL1 has been defined.
```

In order to deploy the paths for all our hosts to all drives in the library (in our case, we have 64 drives), we created a script for generating the Spectrum Protect macro file and then run the macro file using Spectrum Protect administrative command-line client, **dsmadmc**. See Example 3-60.

Example 3-60 Generating the paths for all client nodes

```
root@tsmsrv:/> for i in TSMCL1_STA TSMCL2_STA TSMCL3_STA
do
j=1; while [ $j -le 64 ]; do echo "define path $i GPFSDEVCL${j} srct=server
destt=drive library=GPFSDEVCL device=FILE directory=/tsmgpfs"; let j=j+1; done
done > /tmp/STApath.mac
```

```
##### Run the macro file generated in the above script with dsmadmc

root@tsmsrv:/>dsmadmc -id=admin -pass=admin1234 -itemcommit -noconfirm
-outfile=/tmp/STApath.log macro /tmp/STApath.mac

Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 10/21/14   12:19:53  Last access: 10/21/14   12:13:48
```

3. In the next step, we enable the LAN-free configuration for the client side. The following files show the final configuration for our scenario for the storage agent and Spectrum Protect client option file. For the storage agent configuration of node tsmcl1, the output of the options file (`dsmsta.opt`) and device configuration (`devconfig.txt`) is shown in Example 3-61. Similar configurations apply to the rest of the clients, adapted to their hostnames and IP addresses.

Example 3-61 Configuration files for the storage agent

```
root@tsmcl1:/opt/tivoli/tsm/StorageAgent/bin>cat dsmsta.opt | grep -v "^*"

COMMmethod SHAREDMEM
SHMPort 1510
TXNGroupmax 1024
DEVCONFIG devconfig.txt

SERVERNAME TSMGPFS1

root@tsmcl1:/opt/tivoli/tsm/StorageAgent/bin>cat devconfig.txt
SET STANAME TSMCL1_STA
SET STAPASSWORD 21234c2ebbd16e4d9a5e83b9195a9ab7fc
SET STAHLADDRESS 172.16.20.154
DEFINE SERVER TSMGPFS1 HLADDRESS=172.16.20.153 LLADDRESS=1500
SERVERPA=210a9bc267aee6684d4db46b3de4786a45 SSL=NO
```

The storage agent can be started by using the following script:

`/opt/tivoli/tsm/StorageAgent/bin/rc.tsmstgagnt`

4. The Spectrum Protect client system option file `dsm.sys` in our environment is shown in Example 3-62.

Example 3-62 Spectrum Protect client configuration file

```
SErvername  tsmgpfs1
   COMMMethod         TCPip
   TCPPort            1500
   TCPServeraddress   172.16.20.153
   PASSwordaccess       generate
   Enablelanfree        yes
   LANFREECommmethod    sharedmem
```

5. You can validate the server LAN-free configuration for a node using the Spectrum Protect command: **`validate lan-free`**. See the output of the command in Example 3-63 on page 168 for the node tsmcl1, using the storage agent TSMCL1_STA and the destination storage pool GPFSPOOL.

Example 3-63 Validating the LAN-free configuration

```
tsm: TSMGPFS1>validate lanfree tsmcl1 tsmcl1_sta
ANR0387I Evaluating node TSMCL1 using storage agent TSMCL1_STA for LAN-free data movement.

Node      Storage      Operation     Mgmt Class     Destination      LAN-Free      Explanation
Name      Agent                      Name           Name             capable?
-----     --------     ---------     ----------     ------------     ---------     --------------------
TSMC-     TSMCL1_-     BACKUP        STANDARD       GPFSPOOL         Yes
 L1        STA
TSMC-     TSMCL1_-     ARCHIVE       STANDARD       ARCHIVEPOOL      No            Destination storage
 L1        STA                                                                     pool is DISK.
ANR1706I Ping for server 'TSMCL1_STA' was able to establish a connection.
ANR0388I Node TSMCL1 using storage agent TSMCL1_STA has 1 storage pools capable of LAN-free data
movement and 1 storage pools not capable of LAN-free data movement.
```

Testing the LAN-free transfer

We use the backup-archive client command line (dsmc) on tsmcl1, for testing the data transfer to the Spectrum Scale file system. The tsmcl1 client has a jfs2 file system, `/tsmclient`. In general, depending on the platform and applications used, the source file system can be OS-specific or application-specific (as an example, Oracle RAC on ASM using Spectrum Protect Data Protection).

Example 3-64 shows the output of the **`dsmc selective`** command used to back up the client file system `/tsmclient`.

Example 3-64 Backing up a client file system to Spectrum Scale target pool

```
root@tsmcl1:/>dsmc selective -subdir=yes /tsmclient/
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 0.0
  Client date/time: 10/21/14   15:06:09
(c) Copyright by IBM Corporation and other(s) 1990, 2013. All Rights Reserved.

Node Name: TSMCL1
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 10/21/14   15:06:10  Last access: 10/20/14   15:01:13

Selective Backup function invoked.

Directory-->             262,144 /tsmclient/ [Sent]
Directory-->              16,384 /tsmclient/.snapshots [Sent]
Directory-->               4,096 /tsmclient/TSMCLI_AIX [Sent]
Directory-->               4,096 /tsmclient/TSMSRV [Sent]
Normal File-->    10,737,418,240 /tsmclient/test.10g [Sent]
Directory-->               4,096 /tsmclient/TSMCLI_AIX/usr [Sent]
Normal File-->             3,263 /tsmclient/TSMCLI_AIX/README.1ST [Sent]
.............

Selective Backup processing of '/tsmclient/*' finished without failure.


Total number of objects inspected:        2,779
Total number of objects backed up:        2,779
Total number of objects updated:              0
Total number of objects rebound:              0
```

```
Total number of objects deleted:              0
Total number of objects expired:              0
Total number of objects failed:               0
Total number of bytes inspected:           5.49 GB
Total number of bytes transferred:         5.49 GB
LanFree data bytes:                        5.49 GB
Data transfer time:                       59.35 sec
Network data transfer rate:           97,126.05 KB/sec
Aggregate data transfer rate:         42,805.16 KB/sec
Objects compressed by:                        0%
Total data reduction ratio:                0.00%
Elapsed processing time:                00:02:14
```

Observe in the previous command output the LAN-free transferred bytes at the end of the backup task, confirming that the operation was performed using the LAN-free paths, by writing the data to the Spectrum Protect disk storage pool (Spectrum Scale file system).

Optimizing the Spectrum Scale configuration

The backup/restore jobs have a sequential pattern. Spectrum Scale uses an efficient mechanism to distribute the data on all disks in the file system (striping). The disks in the Spectrum Scale file system should be selected from different arrays (storage level), to take advantage on the Spectrum Scale striping mechanism.

Although Spectrum Scale default values of the file system tuning parameters may cover most of the backup workloads, the following parameters can be considered in your environment:

- File system block size

  By default, the Spectrum Scale file system is created with 256 KB block size, unless otherwise specified on the **mmcrfs** command. For sequential access disk pool (FILE device type), Spectrum Protect uses a block size of 256 KB. The block size parameter also needs to be correlated with array stripe size and the I/O transfer settings in the operating system. See Chapter 2, “Block size and calculation guidelines” on page 51. Consider creating the Spectrum Scale file system with a larger value (1 MB or more). Consider that when set during the file system creation, the block size cannot be modified later.

- pagepool

  The **pagepool** parameter determines the size of the Spectrum Scale file data block cache. A larger value usually provides benefits for most of the sequential workloads. In GPFS 3.5 and later, the default value is 1 GB. The amount of memory available for Spectrum Scale pagepool on a particular node may be restricted by the operating system and other software running on the node. The pagepool is also limited by the memory size of the system. It cannot exceed more than a half of the system memory.

- maxMBpS

  The **maxMBpS** option is an indicator of the maximum throughput in megabytes that can be submitted by Spectrum Scale per second into or out of a single node. It is not a hard limit, rather the maxMBpS value is a hint to Spectrum Scale used to calculate how much I/O can effectively be done for sequential prefetch and write-behind operations. In GPFS 3.5 and later, it defaults to 2048. The maximum value is 100,000.

  The maxMBpS value should be adjusted for the nodes to match the I/O throughput the node is expected to support. For example, you should adjust maxMBpS for nodes that are directly attached to storage. A good rule of thumb is to set maxMBpS to twice the I/O throughput required for a system. For example, if a system has two 8-Gbit HBAs (800 MBps per HBA), maxMBpS should be set to 3200. If the maxMBpS value is set too low, sequential I/O performance may be reduced.

- prefetchThreads

  The **prefetchThreads** parameter controls the maximum possible number of threads dedicated to prefetching data for files that are read sequentially, or to handle sequential write-behind. The default value is 72, which is normally sufficient. You can tune this parameter based on how many prefetchThreads are in use. Use the **mmfsadm** command:

  ```
  mmfsadm dump fs | egrep "nPrefetchThreads:|total wait"
  ```

  The maximum value depends on the sum of worker1Threads + prefetchThreads + nsdMaxWorkerThreads < 1500 on 64-bit architectures.

3.9 Sample Spectrum Scale FPO configuration

This section describes how to configure a Spectrum Scale FPO file system in your cluster.

In this scenario, Linux Spectrum Scale cluster is created by using two nodes and each node is connected to a local storage using SSD and SAS disks. A storage pool named "*fast*" was created to accommodate all the SSD disks, creating a group of high performance devices, and a storage pool named "*slow*" was created to group all the SAS disks. The *slow* storage pool is used to store data that is not dependent on low response times. It is important to determine the type and usage for each physical disk before creating the storage groups.

Storage pools are typically determined by the characteristics of the storage types. In a Spectrum Scale FPO cluster, you have a minimum of two storage pools: one for metadata and data that does not require high performance disks (with standard block allocation), and one for data that is FPO-enabled using high performance disks.

The following steps describe how to set up a new cluster using FPO file systems:

1. To create the Spectrum Scale cluster, a simple input file was defined to describe the node roles and then the **mmcrcluster** command is issued as shown in Example 3-65.

Example 3-65 Sample input file for Spectrum Scale and the cluster creation

```
[root@fpo_linux1 gpfs]# cat gpfs.nodes
fpo_linux1:quorum
fpo_linux2:quorum

[root@fpo_linux1 gpfs]# mmcrcluster -N gpfs.nodes --ccr-enable -C gpfs.FPO -A -r
/usr/bin/ssh -R /usr/bin/scp
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
```

2. Setting the appropriated license is shown in Example 3-66.

Example 3-66 Setting server licenses for the nodes

```
[root@fpo_linux1 gpfs]# mmchlicense fpo--accept -N fpo_linux1,fpo_linux2
The following nodes will be designated as possessing GPFS server licenses:
        fpo_linux1
        fpo_linux2
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
```

```
  affected nodes.  This is an asynchronous process.
mmcrcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. An `fpo_pool` file is created to be used as input file for storage pool definition and also for NSD creation. The file contents are as shown in Example 3-67.

Example 3-67 Input file named fpo_pool

```
[root@fpo_linux1 gpfs]# cat fpo_pool
%pool:
pool=fast
layoutMap=cluster
blocksize=1024K
allowWriteAffinity=yes #this option enables FPO feature
writeAffinityDepth=1 #place 1st copy on disks local to the node writing data
blockGroupFactor=128 #Defines chunk size of 128MB

%pool:
pool=slow
layoutMap=cluster
blocksize=1024K
allowWriteAffinity=yes #this option enables FPO feature
writeAffinityDepth=1 #place 1st copy on disks local to the node writing data
blockGroupFactor=128 #Defines chunk size of 128MB

#Disks in system pool are defined for metadata
 %nsd: nsd=fpo_linux1_NSD_Meta_1 device=/dev/dm-6 servers=fpo_linux1
usage=metadataOnly failureGroup=102 pool=system

#Disks in slow pool
 %nsd: nsd=fpo_linux1_NSD_Data_1 device=/dev/dm-3 servers=fpo_linux1
usage=dataOnly failureGroup=1,0,1 pool=slow
 %nsd: nsd=fpo_linux2_NSD_Data_2 device=/dev/dm-2 servers=fpo_linux2
usage=dataOnly failureGroup=1,0,2 pool=slow

#Disks in fast pool
 %nsd: nsd=fpo_linux1_NSD_Data_3 device=/dev/dm-5 servers=fpo_linux1
usage=dataOnly failureGroup=1,0,1 pool=fast
 %nsd: nsd=fpo_linux2_NSD_Meta_4 device=/dev/dm-3 servers=fpo_linux2
usage=dataOnly failureGroup=1,0,2 pool=fast
```

4. Then, we create the NSDs and the storage pools using the `fpo_pool` file as shown in Example 3-68.

Example 3-68 Creating NSDs

```
[root@fpo_linux1 gpfs]# mmcrnsd -F fpo_pool
mmcrnsd: Processing disk dm-6
mmcrnsd: Processing disk dm-3
mmcrnsd: Processing disk dm-3
mmcrnsd: Processing disk dm-5
mmcrnsd: Processing disk dm-2
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

5. Listing the new NSDs with the **mmlsnsd** command is shown in Example 3-69.

Example 3-69 Listing the new NSDs

```
[root@fpo_linux1 gpfs]# mmlsnsd
File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   fpo_linux1_NSD_Data_1 fpo_linux1
 (free disk)   fpo_linux1_NSD_Data_2 fpo_linux1
 (free disk)   fpo_linux1_NSD_Meta_1 fpo_linux1
 (free disk)   fpo_linux2_NSD_Data_3 fpo_linux2
 (free disk)   fpo_linux2_NSD_Meta_2 fpo_linux2
```

6. Now we can create a new file system. The **mmcrfs** syntax creates a new file system called `fpofs`, automatically mounted (**-A** flag), in all nodes, with one metadata and data copies. The mount point is `/fpofs`. It is using the same input file for the NSDs creation as shown in Example 3-70.

Example 3-70 Creating a new file system using the mmcrfs command

```
[root@fpo_linux1 gpfs]# mmcrfs fpofs -F fpo_pool -A yes -m 2-M 2 -n 32  -r 2-R 2
-S relatime -E no -T /fpofs
The following disks of fpofs will be formatted on node fpo_linux1:
    fpo_linux1_NSD_Meta_1: size 10240 MB
    fpo_linux1_NSD_Data_1: size 10240 MB
    fpo_linux2_NSD_Data_2: size 10240 MB
    fpo_linux1_NSD_Data_3: size 10240 MB
    fpo_linux2_NSD_Meta_4: size 10240 MB
Formatting file system ...
Disks up to size 109 GB can be added to storage pool system.
Disks up to size 103 GB can be added to storage pool fast.
Disks up to size 103 GB can be added to storage pool slow.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Formatting Allocation Map for storage pool fast
Formatting Allocation Map for storage pool slow
Completed creation of file system /dev/fpofs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The **–S** option specifies atime update mode. A value of real time suppresses periodic updating of atime for files and reduces the performance overhead caused due to frequent updates triggered by frequent file read operations.

7. Now we can mount the file system and display the storage pools that are associated with this file system as shown in Example 3-71.

Example 3-71 Mounting file system and displaying storage pool information

```
[root@fpo_linux1 gpfs]# mmmount fpofs -a
Wed Oct 29 16:43:15 EDT 2014: mmmount: Mounting file systems ...

[root@fpo_linux1 gpfs]# mmlspool fpofs
```

```
Storage pools in file system at '/fpofs':
Name                    Id   BlkSize Data Meta Total Data in (KB)   Free Data in
(KB)   Total Meta in (KB)    Free Meta in (KB)
system                   0    256 KB   no  yes              0              0 (
0%)       10485760       10016000 ( 96%)
fast                 65537   1024 KB  yes   no       20971520       20836352 (
99%)              0             0 (  0%)
slow                 65538   1024 KB  yes   no       20971520       20836352 (
99%)              0             0 (  0%)

[root@fpo_linux1 ~]# mmlsdisk fpofs
disk         driver   sector     failure holds    holds
storage
name         type       size       group metadata data  status        availability
pool
------------ -------- ------ ----------- -------- ----- ------------- ------------
------------
fpo_linux1_NSD_Meta_1 nsd       512        102 Yes     No    ready       up
system
fpo_linux1_NSD_Data_1 nsd       512      1,0,1 No      Yes   ready       up
slow
fpo_linux2_NSD_Data_2 nsd       512      1,0,2 No      Yes   ready       up
slow
fpo_linux1_NSD_Data_3 nsd       512      1,0,1 No      Yes   ready       up
fast
fpo_linux2_NSD_Meta_4 nsd       512      1,0,2 No      Yes   ready       up
fast
```

In our scenario, the FPO feature has already been enabled for storage pools *fast* and *slow*, in the storage pool and NSD configuration. The storage pool stanza includes wanted FPO-specific attributes, including the block size for system and data pools.

The following steps are performed to create a policy file and then apply the policy to our file system:

1. Create a file named `fpo_policy` with the following contents.

Example 3-72 Contents of fpo_policy file

```
[root@fpo_linux1 gpfs]# cat fpo_policy
RULE 'perf' SET POOL 'fast' WHERE NAME LIKE '%.tmp' AND setWAD(1)
RULE 'default' SET POOL 'slow'
```

The first rule places all files with extension `.tmp` in *fast* pool, with write-affinity depth set to 1 for these temporary files. The second rule is the default, placing all other files in the *slow* pool using the attributes specified at the storage pool level. This rule applies to all files that do not match any other prior rules and must be specified as the last rule.

2. Now the policy must be installed and enabled by the **mmchpolicy** command. The policies that are currently effective can be listed by using the **mmlspolicy** command as shown in Example 3-73.

Example 3-73 Applying the policy and listing the attributes

```
[root@fpo_linux1 gpfs]# mmchpolicy fpofs fpo_policy -I yes
Validated policy `fpo_policy': parsed 2 Placement Rules, 0 Restore Rules, 0
Migrate/Delete/Exclude Rules,
```

```
        0 List Rules, 0 External Pool/List Rules
Policy `fpo_policy' installed and broadcast to all nodes.

[root@fpo_linux1 gpfs]# mmlspolicy fpofs -L
RULE 'perf' SET POOL 'fast' WHERE NAME LIKE '%.tmp' AND setWAD(1)
RULE 'default' SET POOL 'slow'
```

3. For our test environment, several `.tmp` files and some files without extensions are created in the `fpofs` file system as shown in Example 3-74.

Example 3-74 Listing the files created in the fpofs file system

```
[root@fpo_linux1 fpofs]# ls -ltr
total 264192
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_0.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_1.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_2.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_3.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_4.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_5.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_6.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_7.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_8.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_9.tmp
-rw-r--r--. 1 root root         0 Oct 29 17:20 file_10.tmp
-rw-r--r--. 1 root root 135127040 Oct 29 17:20 file
-rw-r--r--. 1 root root 135127040 Oct 29 17:21 file.tmp
```

4. Now we can execute the **`mmlsattr -L`** command for each individual file to verify whether the placement policy is working as shown in Example 3-75.

Example 3-75 Listing file attributes

```
[root@fpo_linux1 fpofs]# mmlsattr -L /fpofs/file.tmp
file name:            /fpofs/file.tmp
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    fast
fileset name:         root
snapshot name:
Write affinity depth:  1
creation time:        Wed Oct 29 17:21:02 2014
Windows attributes:   ARCHIVE

[root@fpo_linux1 fpofs]# mmlsattr -L /fpofs/file
file name:            /fpofs/file
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    slow
fileset name:         root
snapshot name:
```

```
creation time:        Wed Oct 29 17:20:59 2014
Windows attributes:   ARCHIVE
```

The file with extension `.tmp` is correctly placed in the pool named *fast*, and the file without the extension is placed under the pool named *slow*.

For more information about FPO, see the following resources:

- *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00
- *GPFS V4.1: Administration and Programming Reference,* SA23-1452-00
- *IBM System x Reference Architecture for Hadoop: IBM InfoSphere BigInsights Reference Architecture*, REDP-5009

  http://www.redbooks.ibm.com/abstracts/redp5009.html?Open
- Deploying a big data solution using IBM Spectrum Scale FPO

  http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF

3.10 Integrating with OpenStack Swift

Due to the explosion of unstructured data that is generated by individuals and organizations, a new storage paradigm called *Object Storage* has been developed. Object Storage stores data in a flat namespace that scales to trillions of objects. The design of object storage also simplifies how users access data, supporting new types of applications and allowing users to access data by various methods, including mobile devices and web applications. Data distribution and management are also simplified, allowing greater collaboration across the globe.

OpenStack Swift is an emerging open source Object Storage software platform that is widely used for cloud storage. It is a high-performance and proven product that is used to store the data for thousands of mission-critical commercial installations worldwide. Together, Spectrum Scale and OpenStack Swift provide an enterprise-class object storage solution that efficiently stores, distributes, and retains critical data.

For details about this setup, see the following IBM Redpaper™ publication *A Deployment Guide for IBM Spectrum Scale Object*, REDP-5113:

http://www.redbooks.ibm.com/abstracts/redp5113.html?Open

4

Management and maintenance

This chapter contains the most commonly used information about how to manage your existing Spectrum Scale cluster, what is needed to update or migrate your current Spectrum Scale version, expanding or changing Spectrum Scale cluster configuration. For more information about management and upgrade options, see *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00.

During this chapter, the term General Parallel File System (*GPFS*) is used to mention clusters running GPFS version 3.5 or earlier versions. IBM Spectrum Scale applies only for version 4.1 clusters.

This chapter contains the following topics:

- IBM Spectrum Scale and GPFS migration and update
- Managing IBM Spectrum Scale cluster
- Managing IBM Spectrum Scale file systems
- Managing IBM Spectrum Scale data migration
- Managing the IBM Spectrum Scale network
- Managing IBM Spectrum Scale remote cluster
- Configuring IBM Spectrum Scale callback
- Monitoring IBM Spectrum Scale with SNMPD protocol
- SSH configuration

© Copyright IBM Corp. 2015. All rights reserved.

4.1 IBM Spectrum Scale and GPFS migration and update

This section describes general considerations and tasks to perform when you have to migrate your existing cluster to a new release, or apply fixes when you have a mixed or multicluster environment with different GPFS versions.

For information about how to install GPFS packages, see section 3.4.1, “Installing IBM Spectrum Scale” on page 99.

Note: For more information and the complete migration procedures for Spectrum Scale, see *GPFS V4.1: Concepts, Planning, and Installation Guide*, GA76-0441-00.

4.1.1 GPFS and Spectrum Scale migration considerations

To migrate to Spectrum Scale 4.1 or GPFS 3.1, first consider whether you are migrating from GPFS 3.5, GPFS 3.4, or from an earlier release of GPFS, and then consider coexistence and compatibility issues.

The GPFS supports the limited form of compatibility with earlier versions between two adjacent GPFS releases as described in Table 4-1.

Table 4-1 GPFS upgrade options

Source GPFS version	Target GPFS version	Upgrade option
3.2	3.3	Rolling upgrade
3.3	3.4	Rolling upgrade
3.4	3.5	Rolling upgrade
3.5	4.1	Rolling upgrade
3.4 or 3.3	4.1	No rolling upgrade
3.3 or 3.2	3.5	No rolling upgrade
3.2 or earlier	4.1	Re-create the cluster

This limited compatibility with earlier versions allows you to temporarily operate with a mixture of Spectrum Scale 4.1 and GPFS 3.5 nodes that can be useful for some of the following reasons:

- Within a cluster, this enables you to perform a *rolling upgrade* to the new Spectrum Scale 4.1 version of the code for the nodes that are still running GPFS 3.5. The rolling upgrade allows the administrator to install new Spectrum Scale code one node at a time without shutting down Spectrum Scale on other nodes.
- In a multicluster environment, this allows the individual clusters to be upgraded on their own schedules. Access to the file system data can be preserved even though some of the clusters may still be running GPFS 3.5 level.

Although it is possible to have an environment running different levels of Spectrum Scale, you must upgrade all nodes within a short time. The time dependency exists because some Spectrum Scale 4.1 features become available on each node as soon as the node is upgraded, while other features will not become available until you upgrade all participating nodes. When considering a mixed environment with different GPFS versions, ensure that

your different GPFS versions can coexist. Table 4-2 shows the compatibility between different GPFS versions.

Table 4-2 GPFS compatibility versions

Current GPFS version	Coexistent GPFS versions	Notes
4.1	4.1 and 3.5	All levels and fixes are supported. CCR is only supported on GPFS 4.1
3.5	4.1, 3.5, and 3.4	GPFS 3.5.0.1 cannot coexist with GPFS older than V3.4.0.7
3.4	3.5, 3.4	GPFS 3.4.0.12 cannot coexist with GPFS older than V3.4.0.7. GPFS older than 3.4.0.7 cannot coexist with GPFS 3.5.0.1

When running Spectrum Scale on Windows systems, observe that there is no migration path from Windows Server 2003 R2 (GPFS V3.2.1.5 or later) to Windows Server 2008 (GPFS V3.4 or later). To move your Spectrum Scale Windows 2003 R2 nodes to a supported level of Spectrum Scale, perform the following steps:

1. Remove all the Windows operating system nodes from your cluster.
2. Uninstall GPFS 3.2.1.5 from your Windows operating system nodes. This step is not necessary if you are reinstalling Windows Server 2008 or newer OS from scratch and not upgrading from Server 2003 R2.
3. Install Windows Server 2008 or newer OS and the prerequisites on the nodes.
4. Install a supported level of Spectrum Scale on the Windows Server 2008 or newer OS nodes.
5. Migrate your AIX and Linux nodes from GPFS 3.2.1-5 or later, to a supported level of Spectrum Scale.
6. Add the Windows operating system nodes back to your cluster.

For the latest information about migration, coexistence, and compatibility, see the following source:

Spectrum Scale FAQ in the IBM Knowledge Center:

http://ibm.co/1ysyp8w

Note: Starting with Spectrum Scale 4.1, a full backup (**-t full**) with **mmbackup** is required if a full backup has never been performed with GPFS 3.3 or later.

4.1.2 Migrating to Spectrum Scale 4.1 from GPFS 3.5 (Rolling update)

Spectrum Scale 4.1 supports node-at-a-time migration if the previous nodes in the cluster are running GPFS 3.5. The GPFS 3.5 nodes can coexist and interoperate with nodes that are running Spectrum Scale 4.1. However, some new functions that depend on format changes will not be available until all nodes have been migrated.

To perform a GPFS *rolling update*, use the following steps:

1. Stop all user activity in the file systems on the designated node.

2. Follow any local administrative backup procedures to ensure protection of your file system data if there is a failure.
3. Cleanly unmount the mounted GPFS file system. Do not use force unmount on the designated node. For example:

   ```
   mmumount fsname -N fsaix4
   ```

4. Stop GPFS on the node to be migrated in the cluster, for example:

   ```
   mmshutdown -N fsaix4
   ```

5. Run the appropriate deinstallation program to remove GPFS from the node you are migrating. For example:
   - For Linux nodes:

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```

   - For AIX nodes:

     ```
     installp -u gpfs
     ```

   - For Windows operating system nodes running GPFS 3.5

     Open the “Programs and Features” control panel and remove *IBM General Parallel File System*. And to upgrade a GPFS 3.5 Windows operating system node (SUA based) to GPFS 4.1 Standard Edition (Cygwin based), proceed as follows:
   - Uninstall GPFS and reboot.
   - Uninstall the Spectrum Scale license. Do not uninstall SUA yet, or you may lose Spectrum Scale configuration information.
   - Install Cygwin.
   - Install gpfs.ext-4.1-Windows-license.msi.
   - Install gpfs.ext-4.1.0.x-Windows.msi
   - Install IBM GSKit for Spectrum Scale.
   - Uninstall SUA completely.
6. Copy the installation images and install the Spectrum Scale product on the designated node:
   - For Linux: **`rpm -ivh gpfs*.rpm`**
   - For AIX: **`installp -aXY -d <path_to_install_dir> all`**
7. Build the Spectrum Scale portability layer (Linux only). On Spectrum Scale 4.1, you can use the new tool **`mmbuildgpl`**.
8. Start Spectrum Scale on the designated node in the cluster, for example:

   ```
   mmstartup -N fsaix4
   ```

9. Mount the file systems if this is not done automatically when the Spectrum Scale daemon starts, for example:

   ```
   mmmount fsname -N fsaix4
   ```

10. When all nodes in the cluster have been successfully migrated to the new Spectrum Scale level, proceed to “Completing the GPFS or IBM Spectrum Scale migration” on page 183.

4.1.3 Migrating to Spectrum Scale 4.1 from GPFS 3.4 or GPFS 3.3

Node-at-a-time migration is not available when migrating to Spectrum Scale 4.1 from GPFS 3.4 or GPFS 3.3. The cluster must be completely shut down and all nodes migrated at the same time. If this is not acceptable, and your current level is GPFS 3.4, you might want to consider an intermediate migration to GPFS 3.5 first.

To migrate a cluster to Spectrum Scale 4.1 from GPFS 3.4 or GPFS 3.3, perform these steps:

1. Stop all user activity in the file systems.
2. Follow your backup procedures to ensure protection of your file system data if there is a failure.
3. Cleanly unmount all the mounted GPFS file system. Do not use force unmount, for example:

   ```
   mmumount all -a
   ```

4. Stop GPFS on all nodes in the cluster:

   ```
   mmshutdown -a
   ```

5. Run the appropriate deinstallation program to remove GPFS from all nodes in the cluster. For example:
   - For Linux nodes:

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```

   - For AIX nodes:

     ```
     installp -u gpfs
     ```

   - For Windows operating system nodes running GPFS 3.4

     Open the “Programs and Features” control panel and remove *IBM General Parallel File System*. And to upgrade a GPFS 3.4 Windows operating system node (SUA based) to Spectrum Scale 4.1 Standard Edition (Cygwin based), proceed as follows:
   - Uninstall GPFS and reboot.
   - Uninstall the GPFS license. Do not uninstall SUA yet, or you may lose GPFS configuration information.
   - Install Cygwin.
   - Install gpfs.ext-4.1-Windows-license.msi.
   - Install gpfs.ext-4.1.0.x-Windows.msi
   - Install IBM GSKit for Spectrum Scale.
   - Uninstall SUA completely.
6. Copy the installation images and install the new Spectrum Scale version on each node in the cluster:
   - For Linux: **rpm -ivh gpfs*.rpm**
   - For AIX: **installp -aXY -d <path_to_install_dir> all**
7. Build the Spectrum Scale portability layer (Linux only). On Spectrum Scale 4.1, you can use the new tool **mmbuildgpl**.
8. Start Spectrum Scale on all nodes in the cluster:

   ```
   mmstartup -a
   ```

9. Mount the file systems if this is not done automatically when the Spectrum Scale daemon starts.

   ```
   mmmount all -a
   ```

10. Verify if all your Spectrum Scale file systems were mounted.

    ```
    mmlsmount all -L
    ```

11. Proceed to “Completing the GPFS or IBM Spectrum Scale migration” on page 183.

4.1.4 Migrating to Spectrum Scale 4.1 from GPFS 3.2 or earlier releases of GPFS

If you are running GPFS versions that do not support direct migration to 4.1, you can consider an intermediate migration to GPFS 3.5, or any other supported release, first.

To migrate your GPFS cluster to Spectrum Scale 4.1 from GPFS 3.2 or earlier, follow these steps:

1. Stop all user activity in the file systems.
2. Follow any local administrative backup procedures to ensure protection of your file system data if there is a failure.
3. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.

   ```
   mmumount all -a
   ```

4. Shut down the GPFS daemon on all nodes in the cluster:

   ```
   mmshutdown -a
   ```

5. Export the GPFS file systems by issuing the **mmexportfs** command:

   ```
   mmexportfs all -o exportDataFile
   ```

   This command creates the configuration output file `exportDataFile`, which contains all exported configuration data. Retain this file because it is required when issuing the **mmimportfs** command to import your file systems into the new cluster or if you decide to go back to the previous release.
6. Record any site-specific configuration attributes that are currently in effect and that you want to preserve. You can use **mmlsconfig** output.
7. Delete all existing nodes by issuing this command:

   ```
   mmdelnode -a
   ```

8. Run the appropriate deinstallation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```

   - For AIX nodes:

     ```
     installp -u gpfs
     ```

   - For Windows operating system nodes

   Open the “Programs and Features” control panel and remove *IBM General Parallel File System*.
9. Copy the installation images and install the Spectrum Scale product on each node in the cluster.
10. Decide which nodes in your system are quorum nodes.
11. Create a Spectrum Scale cluster across all wanted nodes by issuing the **mmcrcluster** command. Run the **mmchlicense** command to set the appropriate licenses after the cluster is created.
12. Using the **mmchconfig** command, restore your site-specific configuration attributes from step 6.

13. To complete the movement of your file systems to the new cluster, using the configuration file created in step 5, issue:

    ```
    mmimportfs all -i exportDataFile
    ```

14. Start Spectrum Scale on all nodes in the new cluster:

    ```
    mmstartup -a
    ```

15. Mount the file systems if this is not done automatically when the Spectrum Scale daemon starts.

    ```
    mmmount all -a
    ```

16. Verify if all the file systems were mounted

    ```
    mmlsmount all -L
    ```

17. Proceed to “Completing the GPFS or IBM Spectrum Scale migration” on page 183.

Note: For Windows operating system migrations, it is required that all Spectrum Scale administration commands are issued from the Spectrum Scale node that is running the newer version.

4.1.5 Completing the GPFS or IBM Spectrum Scale migration

When the new level of Spectrum Scale is satisfactory for your environment, you must complete migration of both the cluster configuration data and all file systems. If you decide not to migrate, you can revert to the previous level of Spectrum Scale (refer to the section *Reverting to the previous level of GPFS* in the IBM *GPFS V4.1: Concepts, Planning, and Installation Guide*, GA76-0441-00).

After you have migrated *all* nodes to the latest Spectrum Scale licensed program, execute the following steps:

1. Migrate the cluster configuration data and enable new cluster-wide functionality:

   ```
   mmchconfig release=LATEST
   ```

   The **mmchconfig** command lists the names of the nodes that are not available or cannot be reached. If this is the case, correct the problem and reissue the command until all nodes can be verified and the command completes successfully. After issuing the **mmchconfig release=LATEST** command, the **mmlsconfig** output should be similar to what is shown in Example 4-1.

Example 4-1 The mmlsconfig output after setting the release to LATEST

```
root@fsaix4:/var/adm/ras>mmlsconfig
Configuration data for cluster tsmgpfs.tsmsrv:
----------------------------------------------
clusterName tsmgpfs.tsmsrv
clusterId 12742445374757761618
autoload yes
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks tsmdisk1
prefetchThreads 100
adminMode central

File systems in cluster tsmgpfs.tsmsrv:
```

```
---------------------------------------
/dev/tsmclient
/dev/tsmgpfs
```

Note that the ***minReleaseLevel*** shows the minimum supported Spectrum Scale version that can be used within the cluster so be careful when issuing the **mmchconfig release=LATEST** because once the command is executed, the cluster will not support any node running lower GPFS versions than specified by the ***minReleaseLevel***.

2. After the **mmchconfig** command, you can assign an appropriate Spectrum Scale license to each of the nodes in the cluster:
   - To see what the minimum required Spectrum Scale license is for each of the nodes in the cluster, issue:

     ```
     mmlslicense -L
     ```

   - To assign a Spectrum Scale server license to the nodes that require it, issue:

     ```
     mmchlicense server -N NodeList
     ```

   - To assign a Spectrum Scale client license to the nodes that require it, issue:

     ```
     mmchlicense client -N NodeList
     ```

3. Enable backward-compatible format changes or migrate all file systems to the latest metadata format changes.

   Attention: Before continuing with this step, it is important to understand the differences between **mmchfs -V compat** and **mmchfs -V full**:

   - If you issue the **mmchfs -V compat** command, only changes that are backward compatible with GPFS 3.5 are enabled. Nodes in remote clusters that are running GPFS 3.5 will still be able to mount the file system. Nodes running GPFS 3.4 or earlier will no longer be able to mount the file system.
   - If you issue the **mmchfs -V full** command, all new functions that require different on-disk data structures are enabled. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that has the file system mounted at the time this command is issued, the **mmchfs** command fails.
   - To enable backward-compatible format changes, issue the following command:

     ```
     mmchfs FileSystem -V compat
     ```

   - To migrate all file systems to the latest metadata format changes, issue the following command:

     ```
     mmchfs FileSystem -V full
     ```

     Certain new file system features might require additional processing that cannot be handled by the **mmchfs -V** command alone. To fully activate such features, in addition to **mmchfs -V**, you will also have to run the **mmmigratefs** command. An example of such a feature is enabling fast extended attributes for file systems. This option is not applied to GPFS file systems older than GPFS 3.4.
   - To activate fast extended attributes, issue the following command:

     ```
     mmmigratefs FileSystem --fastea
     ```

     More information about fastea is in 4.3.10, “Optimizing extended attributes: The fastea option” on page 207.

4. If you use the **mmbackup** command to back up your file system, a full backup might be required if a full backup has never been performed with GPFS 3.3 or later.

Note: For the procedures about how to revert your Spectrum Scale version to previous GPFS levels, see *GPFS V4.1: Concepts, Planning, and Installation Guide*, GA76-0441-00 under the "*Reverting to the previous level of GPFS*" section.

4.1.6 Applying corrective fixes to IBM Spectrum Scale

When you are planning to apply corrective fixes (*Program Temporary Fixes* or PTFs) in your Spectrum Scale environment, ensure that you are using the correct packages for your specific Spectrum Scale version. When installing corrective fixes, your Spectrum Scale version will not be changed, only the level will be changed, for example, as from 3.5.0.0 to 3.5.0.6, or from 4.1.0.0 to 4.1.0.4.

Note: Sometimes IBM can ask you to install *fixes*. Follow the specific instructions for fix installation that will be sent with the fix package.

To download the installation fixes, go to the IBM Spectrum Scale support website:

`http://ibm.co/1yTASdC`

Reviewing the Spectrum Scale FAQs for the latest service information might help when downloading the fixes:

`http://ibm.co/1fRXoxg`

Ensure that you read the `readme` file. This file indicates which level the downloaded update is for and what you have to do, if anything, before applying the update.

As with migration of the latest level of Spectrum Scale, applying fixes is done one node at a time, so your cluster can still work. Of course when a node is stopped, be aware of the cluster quorum availability.

You must follow several rules before installing the fixes, such as:

1. Every application that uses Spectrum Scale data must be closed on the node where you are applying fixes. If the file system is an NFS-exported file system, the file system must be unexported from NFS. The file system must be unmounted.
2. Spectrum Scale must be stopped with the **mmshutdown** command.
3. Ensure that the kernel extensions have been unloaded with the **mmfsenv -u** command.
4. Then, you can proceed to install fixes for the specific operating system.

Note: Windows operating system version has no upgrade option. Instead, you have to uninstall the previous version and install the most recent version on your node.

In the following steps, we show examples on how to update a Spectrum Scale 4.1.0.0 AIX node to Spectrum Scale 4.1.0.4. The fixes were already downloaded and the Spectrum Scale node name is *fsaix4*.

1. Verifying the current installed version.

   The **lslpp** command shows the fixes that are installed on a Spectrum Scale 4.1 node as shown in Example 4-2 on page 186.

Example 4-2 Verifying current Spectrum Scale level on an AIX node

```
root@fsaix4:/var/adm/ras>lslpp -l |grep gpfs
  gpfs.base                  4.1.0.0  COMMITTED  GPFS File Manager
  gpfs.crypto                4.1.0.0  COMMITTED  GPFS Cryptographic Subsystem
  gpfs.ext                   4.1.0.0  COMMITTED  GPFS Extended Features
  gpfs.gskit                8.0.50.32  COMMITTED  GPFS GSKit Cryptography
  gpfs.base                  4.1.0.0  COMMITTED  GPFS File Manager
```

2. Shutting down the Spectrum Scale on the current node as shown in Example 4-3.

Example 4-3 Stopping the Spectrum Scale services

```
root@fsaix4:/var/adm/ras>mmshutdown
Mon Oct 20 19:31:22 EDT 2014: mmshutdown: Starting force unmount of GPFS file
systems
Mon Oct 20 19:31:27 EDT 2014: mmshutdown: Shutting down GPFS daemons
Shutting down!
Mon Oct 20 19:31:30 EDT 2014: mmshutdown: Finished
```

3. Unloading the Spectrum Scale kernel extension with the **mmfsenv -u** command as shown in Example 4-4.

Example 4-4 The mmfsenv -u command

```
root@fsaix4:/var/adm/ras>mmfsenv -u
/usr/lpp/mmfs/bin/mmfskxload: /usr/lpp/mmfs/bin/mmfs is not loaded.
```

4. Applying Spectrum Scale fixes by **smitty update_all**. Make sure the option "*ACCEPT new license agreements*?" is set to *YES.* The update process output should be similar as shown in Example 4-5.

Example 4-5 smitty update_all results

```
Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------------
gpfs.base                   4.1.0.4         USR         APPLY       SUCCESS
gpfs.base                   4.1.0.4         ROOT        APPLY       SUCCESS
gpfs.ext                    4.1.0.4         USR         APPLY       SUCCESS
gpfs.crypto                 4.1.0.4         USR         APPLY       SUCCESS
```

5. Check that the **lslpp -l** command gives you the new Spectrum Scale level as shown in Example 4-6.

Example 4-6 The lslpp -l |grep gpfs after the update

```
root@fsaix4:/var/adm/ras>lslpp -l |grep gpfs
  gpfs.base                  4.1.0.4  COMMITTED  GPFS File Manager
  gpfs.crypto                4.1.0.4  COMMITTED  GPFS Cryptographic Subsystem
  gpfs.ext                   4.1.0.4  COMMITTED  GPFS Extended Features
  gpfs.gskit                8.0.50.32  COMMITTED  GPFS GSKit Cryptography
  gpfs.base                  4.1.0.4  COMMITTED  GPFS File Manager
```

6. After applying the fixes, you can start the Spectrum Scale with the following command:

```
mmstartup
```

7. Verify if Spectrum Scale has started and activated correctly by using the **mmgetstate -aL** command as shown in Example 4-7.

Example 4-7 The mmgetstate command

```
root@tsmsrv:/usr/lpp/mmfs/bin>mmgetstate -aL
Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
1      tsmsrv             1        2          4       active      quorum node
2      tsmcl1             1        2          4       active      quorum node
3      tsmcl2             1        2          4       active
4      fsaix4             1        2          4       active
```

8. Mount all Spectrum Scale file systems that are not mounted automatically with the command:

```
mmmount all
```

4.2 Managing IBM Spectrum Scale cluster

This section describes some of the most common management tasks that Spectrum Scale cluster administrators can perform daily. For a full and detailed list of all management operations available in a Spectrum Scale cluster, see *GPFS V4.1: Administration and Programming Reference,* SA23-1452-00.

This section covers the following administration tasks:

- Managing cluster repository
- Managing IBM Spectrum Scale nodes

4.2.1 Managing cluster repository

Spectrum Scale commands store configuration and file system information in one or more files collectively known as *Spectrum Scale cluster configuration data files*. These files are not intended to be modified manually.

The Spectrum Scale administration commands are designed to keep these files synchronized between each other and with the Spectrum Scale system files on each node in the cluster. The Spectrum Scale commands constantly update the Spectrum Scale cluster configuration data files, and any user modification made to this information may be lost without warning. On AIX nodes, this includes the Spectrum Scale file system stanzas in `/etc/filesystems` and on Linux nodes, the lists in `/etc/fstab`.

When using traditional repository type, the Spectrum Scale cluster configuration data is stored in the `/var/mmfs/gen/mmsdrfs` file and this file is stored on the nodes designated as the primary Spectrum Scale cluster configuration server, and if specified, the secondary Spectrum Scale cluster configuration server.

Before Spectrum Scale 4.1, when running Spectrum Scale administration commands, it is necessary for the Spectrum Scale cluster configuration data to be accessible to the node running the command. Commands that update the `mmsdrfs` file require that both the primary and, if specified, the secondary Spectrum Scale cluster configuration server nodes are accessible. If one of the cluster configuration server nodes is inaccessible, you can designate a new primary or secondary cluster configuration servers by using the **mmchcluster**

command. Similarly, when the Spectrum Scale daemon starts, at least one of the two server nodes must be accessible.

The new method of storing configuration data, also called *CCR*, is the default method for new clusters created on Spectrum Scale 4.1. Existing clusters can be converted to the new repository type using the *–ccr-enable* option of the **mmchcluster** command.

Using CCR has the advantage that full read/write access to the configuration data remains available as long as a majority of quorum nodes are accessible. When using CCR, the concept of primary and secondary nodes no longer exists. For example, in a two-node cluster with tiebreaker disks, it is still possible to run commands that change the `mmsdrfs` file if one of the two nodes has failed, as long as the surviving node has access to the tiebreaker disks.

Verifying if CCR is enabled

In Example 4-8, we are using a Spectrum Scale Linux only cluster. To list if the Spectrum Scale cluster has *ccr* enabled, run the **mmlscluster** command.

Example 4-8 Using mmlscluster to identify CCR enablement

```
[root@fslinux2 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS.Spectrum
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    fslinux2
  Secondary server:  fslinux1

 Node  Daemon node name             IP address     Admin node name  Designation
--------------------------------------------------------------------------------
   1   fslinux1                    172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                    172.16.20.158  fslinux2         quorum-manager
```

After issuing the **mmlscluster** command, check the **Repository type** parameter. If the output is "*server-based*," it means that you are not using CCR.

Enabling CCR

To enable the CCR repository, use the **mmchcluster --ccr-enable** command. When enabling CCR, there is no need to stop the cluster. Example 4-9 shows an active cluster that is not using CCR.

Example 4-9 Displaying an active cluster without CCR

```
[root@fslinux2 ~]# mmgetstate -aL

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      fslinux1           1        2          2       active      quorum node
       2      fslinux2           1        2          2       active      quorum node
```

```
[root@fslinux2 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS.Spectrum
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    fslinux2
  Secondary server:  fslinux1

 Node  Daemon node name             IP address     Admin node name  Designation
--------------------------------------------------------------------------------
   1   fslinux1                    172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                    172.16.20.158  fslinux2         quorum-manager
```

To change the cluster to CCR enabled, use the **mmchcluster** command as shown in Example 4-10.

Example 4-10 Enabling CCR with mmchcluster

```
[root@fslinux1 ~]#  mmchcluster --ccr-enable
mmchcluster: Command successfully completed
mmchcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

After the **mmchcluster** command, you can verify the cluster status and the output should be as shown in Example 4-11.

Example 4-11 Listing cluster status after enabling CCR

```
[root@fslinux2 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS.Spectrum
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name             IP address     Admin node name  Designation
--------------------------------------------------------------------------------
   1   fslinux1                    172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                    172.16.20.158  fslinux2         quorum-manager
```

Note that the *Repository type* attribute now shows CCR and there are no definitions about primary or secondary servers.

Disabling CCR

It is possible to disable the CCR and revert the cluster to the traditional primary or backup configuration server semantics, but all nodes must be shut down before disabling CCR.

To disable the CCR, use the **mmchcluster --ccr-disable** command after shutting down the nodes as shown in Example 4-12.

Example 4-12 Verifying cluster status and disabling CCR

```
[root@fslinux1 /]# mmgetstate -aL

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      fslinux1           0        0          2       down        quorum node
       2      fslinux2           0        0          2       down        quorum node

[root@fslinux1 /]#  mmchcluster --ccr-disable
Verifying GPFS is stopped on all nodes ...
mmchcluster: GPFS cluster configuration servers:
mmchcluster:   Primary server:    fslinux2
mmchcluster:   Secondary server:  fslinux1
mmchcluster: Command successfully completed
```

After the **mmchcluster** command, verify whether the CCR is disabled as shown in Example 4-13.

Example 4-13 Verifying cluster status after disabling the CCR

```
[root@fslinux1 /]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS.Spectrum
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    fslinux2
  Secondary server:  fslinux1

 Node  Daemon node name             IP address     Admin node name  Designation
-------------------------------------------------------------------------------
   1   fslinux1                    172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                    172.16.20.158  fslinux2         quorum-manager
```

Now the cluster has the “Repository Type” set as “server-based”.

4.2.2 Changing cluster manager nodes

In the default Spectrum Scale cluster manager configuration, a quorum node can become the cluster manager even if it is not designated as a “*manager*” node. In some very specific environments, it might not be acceptable to have the cluster manager in a non-manager node

because it could lead to unexpected behavior in existing clusters, for example, in setups with not enough reliable nodes configured as manager nodes.

Starting in Spectrum Scale 4.1, it is possible to change the default behavior of the cluster to allow you to explicitly define which nodes can become the cluster manager nodes. A new attribute called "*clusterManagerSelection*" was added to the cluster configuration and can have the following values:

- **quorumNode**

  Selection behaves as default: Any quorum node is eligible to become a cluster manager

- **preferManager**

  Choose the lowest-numbered manager node if one is reachable (default option on 4.1 clusters)

- **enforceManager**

  Strict manager mode: Only manager nodes can be selected, and non-manager nodes will not attempt to run the election

The variable can only be changed once the cluster is operating at the 4.1 level and the cluster must be down for the variable to be changed with the **mmchconfig** command.

To list this new variable, execute the **mmlsconfig** command as shown in Example 4-14.

Example 4-14 Using the mmlsconfig to display the clusterManagerSelection attribute

```
[root@fslinux1 ~]# mmlsconfig clusterManagerSelection
clusterManagerSelection preferManager
```

To change the default attribute to another value, ensure that the cluster is not running (all Spectrum Scale nodes down) and perform the following command as shown in Example 4-15.

Example 4-15 Changing the clusterManagerSelection attribute with mmchconfig

```
[root@fslinux1 ~]# mmchconfig clusterManagerSelection=quorumNode
Verifying GPFS is stopped on all nodes ...
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

After the **mmconfig** command is performed, verify the cluster configuration by displaying the current settings and restarting the cluster as shown in Example 4-16.

Example 4-16 Displaying the new clusterManagerSelection attribute with mmlsconfig

```
[root@fslinux1 ~]# mmlsconfig clusterManagerSelection
clusterManagerSelection quorumNode

[root@fslinux1 ~]# mmstartup -a
Thu Oct 23 11:04:32 EDT 2014: mmstartup: Starting GPFS ...
```

4.2.3 Managing IBM Spectrum Scale nodes

Usually the most common actions regarding Spectrum Scale nodes are either addition of a new node or the deletion of an existing node. In this section, we describe how to add a node and how to delete nodes from your existing Spectrum Scale cluster.

Adding a node to the cluster

One of the benefits that Spectrum Scale provides is that you have a highly scalable cluster. You can start with a small cluster and add nodes to your cluster.

To add a new node to your Spectrum Scale cluster, use the **mmaddnode** command. When adding a node, you can specify the node name (host name), the TCP/IP interface that Spectrum Scale will use to contact all nodes, and the node designation. A Spectrum Scale node can be either a manager or a client node, as follows:

- **manager | client**

  Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is client.

- **quorum | nonquorum**

  Indicates whether a node is counted as a quorum node. The default is nonquorum.

  Before Spectrum Scale 4.1, if you are designating a new node as a quorum node, and adminMode central is in effect for the cluster, Spectrum Scale must be down on all nodes in the cluster. This restriction no longer applies when using Spectrum Scale 4.1. Alternatively, you can choose to add the new nodes as nonquorum, and after Spectrum Scale has been successfully started on the new nodes, you can change their designation to quorum using the **mmchnode** command.

  Usually the **mmaddnode** command is as follows:

  ```
  mmaddnode -N node1:quorum-manager,node2:quorum
  ```

  In this example, we added a node called *node1* as a quorum-manager, and *node2* as a quorum client.

Adhere to the following rules when adding nodes to a Spectrum Scale cluster:

- You can issue the command only from a node that already belongs to the Spectrum Scale cluster.
- Although a node can mount file systems from multiple clusters, the node itself can only be added to a single cluster using the **mmcrcluster** or the **mmaddnode** command.
- The nodes must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and reissue the command to add those nodes.
- After the nodes are added to the cluster, use the **mmchlicense** command to designate appropriate Spectrum Scale licenses to the new nodes.
- Ensure that the network is properly configured on the new nodes so that both existing and new nodes can communicate to each other. Check whether your `/etc/hosts` entries are updated on every node or on DNS if you are using one.
- The SSH must be working without password-request among all nodes.

Note: If your cluster is configured as admin-mode=central, the SSH passwordless access is not required.

In the ITSO lab environment, we have the following Spectrum Scale cluster running. See Example 4-17.

Example 4-17 The ITSO lab Spectrum Scale cluster

```
GPFS cluster information
========================
  GPFS cluster name:         tsmgpfs.tsmsrv
  GPFS cluster id:           12742445374757761618
  GPFS UID domain:           tsmgpfs.tsmsrv
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1   tsmsrv            172.16.20.153  tsmsrv           quorum-manager
   2   tsmcl1            172.16.20.154  tsmcl1           quorum-manager
   3   tsmcl2            172.16.20.155  tsmcl2
```

The node to be added is called *"fsaix4"*. The node used to run the configuration process is *tsmsrv.* Before running any Spectrum Scale command, you need to ensure that the network and SSH configuration is complete:

1. Ensure that the `/etc/hosts` file on all nodes is up to date, including the information about the new host.
2. On the fsaix4 node, run the **`ssh-keygen -t rsa`** command to have proper SSH configuration files, as shown in Example 4-18.

Example 4-18 Using the ssh-keygen command to create new SSH keys

```
root@fsaix4:/.ssh>ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (//.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in //.ssh/id_rsa.
Your public key has been saved in //.ssh/id_rsa.pub.
The key fingerprint is:
d4:9b:e1:5b:d1:b1:e0:b7:54:cf:49:87:92:59:0a:d4 root@fsaix4
The key's randomart image is:
+--[ RSA 2048]----+
|          .o..+ooo|
|          . oE+o=+|
|         . o +.=.o|
|        . . + + . |
|         S + . .  |
|            o     |
|           .      |
|                  |
|                  |
+-----------------+
```

3. On node *fsaix4*, a new `id_rsa.pub` file was created. In order to enable ssh-passwordless communication, the SSH requires the `$HOME/.ssh/authorized_keys` file. This file must contain the public keys of all nodes. To fill the `authorized_keys` file, perform the following steps:

   a. On node *fsaix4*, copy the `id_rsa.pub` public key to `id_rsa.`**"**`nodename`**".pub** file. The file name now is `id_rsa.fsaix4.pub`.

   b. Send the `id_rsa.fsaix4.pub` file to node *tsmsrv* using **ftp** or **scp** and ensure that the file is under the `$HOME/.ssh` directory as shown in Example 4-19.

Example 4-19 Transferring the public ssh keys

```
root@fsaix4:/.ssh>scp id_rsa.fsaix4.pub tsmsrv:/.ssh
root@tsmsrv's password:
id_rsa.fsaix4.pub 100%  393     0.4KB/s   00:00
```

   c. The node tsmsrv already belongs to the Spectrum Scale cluster, therefore a file named `authorized_keys` exists. At this point, you need to append the `id_rsa.fsaix4.pub` file over the existing `authorized_keys`. On node *fsaix4:*

```
cat id_rsa.fsaix4.pub >> authorized_keys
```

   d. When this process is complete, you can transfer the `authorized_keys` file from the manager node (*tsmsrv* in this example) for every node as shown in Example 4-20.

Example 4-20 Transferring the authorized_keys to the nodes

```
root@tsmsrv:/.ssh>scp authorized_keys fsaix4:/.ssh
root@fsaix4's password:
authorized_keys                               100% 1179     1.2KB/s   00:00
```

   When done, use the **ssh *nodename* date** command to test whether the SSH configuration works without asking for the password.

4. The SSH configuration is now complete and we can add the node to the cluster. If you need to install the Spectrum Scale software in your environment, proceed to installation before continuing.

5. Adding the nodes to the Spectrum Scale cluster:

   a. From node tsmsrv, add node fsaix4 as shown in Example 4-21.

Example 4-21 The mmaddnode command

```
root@tsmsrv:/.ssh>mmaddnode -N fsaix4
Tue Oct 21 15:28:31 EDT 2014: mmaddnode: Processing node fsaix4
mmaddnode: Command successfully completed
mmaddnode: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmaddnode: Propagating the cluster configuration data to all affected nodes.
```

   b. As suggested by the command output, designate a proper Spectrum Scale license for the new configured node as shown in Example 4-22.

Example 4-22 Adding a license to the new node

```
root@tsmsrv:/.ssh>mmchlicense server --accept -N fsaix4
The following nodes will be designated as possessing GPFS server licenses:
        fsaix4
mmchlicense: Command successfully completed
```

```
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process..
```

The new node is configured in the cluster and is now a member of the Spectrum Scale cluster.

6. On node *fsaix4*, start the Spectrum Scale cluster with the following command:

   ```
   mmstartup
   ```

7. Verify the status of the new Spectrum Scale node with the **mmgetstate** command as shown in Example 4-23.

Example 4-23 The node now is active

```
root@tsmsrv:/.ssh>mmgetstate -aL
Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
1      tsmsrv             1        2          4       active      quorum node
2      tsmcl1             1        2          4       active      quorum node
3      tsmcl2             1        2          4       active
4      fsaix4             1        2          4       active
```

Removing a node from the cluster

In certain situations, you might need to delete your existing node from the cluster, for operating system maintenance or even if you plan to change the existing nodes for nodes on new hardware. You can delete nodes from a Spectrum Scale cluster by issuing the **mmdelnode** command.

You must follow these rules when deleting nodes:

- A node being deleted cannot be the primary or secondary Spectrum Scale cluster configuration server unless you intend to delete the entire cluster. Verify this by issuing the **mmlscluster** command. If a node to be deleted is one of the servers and you intend to keep the cluster, issue the **mmchcluster** command to assign another node as a configuration server before deleting the node.
- A node that is being deleted cannot be designated as an NSD server for any disk in the Spectrum Scale cluster, unless you intend to delete the entire cluster. Verify this by issuing the **mmlsnsd** command. If a node that is to be deleted is an NSD server for one or more disks, move the disks to nodes that will remain in the cluster. Issue the **mmchnsd** command to assign new NSD servers for those disks.
- Spectrum Scale must be shut down on the nodes being deleted. Issue the **mmshutdown** command before deleting the node.

In the IBM ITSO lab environment, the following Spectrum Scale cluster is used as shown in Example 4-24.

Example 4-24 The IBM ITSO lab environment for deleting a node

```
GPFS cluster name:         tsmgpfs.tsmsrv
  GPFS cluster id:           12742445374757761618
  GPFS UID domain:           tsmgpfs.tsmsrv
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
-------------------------------------------------------------------
```

```
   1   tsmsrv            172.16.20.153  tsmsrv           quorum-manager
   2   tsmcl1            172.16.20.154  tsmcl1           quorum-manager
   3   tsmcl2            172.16.20.155  tsmcl2
   4   fsaix4            172.16.20.156  fsaix4
```

To delete an existing Spectrum Scale node called *fsaix4*, follow these steps:

1. Ensure that you are logged in the node that will be deleted and no application that might be using Spectrum Scale is running and stop the Spectrum Scale services:

   ```
   mmshutdown -N fsaix4
   ```

2. After stopping the cluster on the node *fsaix4*, the **mmdelnode** command is executed from one of the quorum-manager nodes (*tsmsrv* node as shown in Example 4-25).

Example 4-25 Deleting the Spectrum Scale node with the mmdelnode command

```
root@tsmsrv:/gpfs>mmdelnode -N fsaix4
Verifying GPFS is stopped on all affected nodes ...
mmdelnode: Command successfully completed
mmdelnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. Execute the **mmlscluster** command to check that the node was deleted from the Spectrum Scale configuration as shown in Example 4-26.

Example 4-26 mmlscluster after deleting the node with the mmdelnode command

```
root@tsmsrv:/gpfs>mmlscluster
GPFS cluster information
========================
  GPFS cluster name:         tsmgpfs.tsmsrv
  GPFS cluster id:           12742445374757761618
  GPFS UID domain:           tsmgpfs.tsmsrv
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1   tsmsrv            172.16.20.153  tsmsrv           quorum-manager
   2   tsmcl1            172.16.20.154  tsmcl1           quorum-manager
   3   tsmcl2            172.16.20.155  tsmcl2
```

4.3 Managing IBM Spectrum Scale file systems

Some file system management tasks might be required to maintain a Spectrum Scale cluster. In this topic, some of the most used file system management tasks are covered such as:

- Listing file systems
- Mounting a file system
- Unmounting a file system
- Creating an IBM Spectrum Scale file system
- Removing a file system
- Repairing a file system
- Reducing file system fragmentation

- Listing file system attributes
- Changing file system attributes
- Optimizing extended attributes: The fastea option

4.3.1 Listing file systems

In order to list the existing Spectrum Scale file systems, some commands can be used to show different and specific information such as **mmlsconfig**, **mmlsnsd**, and **mmlsmount** commands.

The mmlsconfig command

This command shows basic information about the cluster and the file systems that are created. See Example 4-27.

Example 4-27 The mmlsconfig output

```
root@tsmsrv:/.ssh>mmlsconfig
Configuration data for cluster tsmgpfs.tsmsrv:
----------------------------------------------
clusterName tsmgpfs.tsmsrv
clusterId 12742445374757761618
autoload yes
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks tsmdisk1
prefetchThreads 100
adminMode central

File systems in cluster tsmgpfs.tsmsrv:
---------------------------------------
/dev/gpfs2
/dev/tsmclient
/dev/tsmgpfs
```

The mmlsnsd command

The **mmlsnsd** command shows the file systems and the respective NSDs that are being used by these file systems. Example 4-28 shows the default **mmlsnsd** output.

Example 4-28 The mmlsnsd command

```
root@tsmsrv:/.ssh>mmlsnsd

 File system   Disk name    NSD servers
------------------------------------------------------------------------------
 gpfs2         gpfs1nsd     (directly attached)
 gpfs2         gpfs2nsd     (directly attached)
 gpfs2         gpfs3nsd     (directly attached)
 gpfs2         gpfs4nsd     (directly attached)
 tsmclient     migratensd   (directly attached)
 tsmgpfs       tsmdisk1     (directly attached)
 tsmgpfs       tsmdisk2     (directly attached)
 tsmgpfs       tsmdisk3     (directly attached)
 tsmgpfs       tsmdisk4     (directly attached)
```

The mmlsmount command

The **mmlsmount** command shows all the Spectrum Scale file systems and the nodes that are mounting the file systems. The **mmlsmount** command can be used to identify nodes that are experiencing mounting problems. The output should be similar to Example 4-29.

Example 4-29 The mmlsmount command

```
root@tsmsrv:/.ssh>mmlsmount all -L

File system gpfs2 is mounted on 3 nodes:
  172.16.20.154   tsmcl1
  172.16.20.153   tsmsrv
  172.16.20.156   fsaix4

File system tsmclient is mounted on 4 nodes:
  172.16.20.154   tsmcl1
  172.16.20.153   tsmsrv
  172.16.20.155   tsmcl2
  172.16.20.156   fsaix4

File system tsmgpfs is mounted on 4 nodes:
  172.16.20.153   tsmsrv
  172.16.20.155   tsmcl2
  172.16.20.154   tsmcl1
  172.16.20.156   fsaix4
```

The **mmlsmount** command with the options **all** and **-L** shows a detailed view for all Spectrum Scale file systems.

4.3.2 Mounting a file system

You must explicitly mount a Spectrum Scale file system if this is the first time the file system is being mounted after its creation, or you specified not to automatically mount (-A no) the file system when you created it.

If you allowed the default value for the automatic mount option (-A yes) when you created the file system, you do not need to use this procedure after starting Spectrum Scale on the nodes because the **mmstartup** command mounts all the file systems automatically.

To mount a Spectrum Scale file system, enter:

```
mmmount device
```

Where *device* is the name of the file system. For example, to mount the file system gpfs2, enter:

```
mmmount gpfs2
```

When mounting a file system, the **mmmount** command can be used to mount file systems locally or in remote nodes. Following are the valid mounting options:

```
mmmount {Device | DefaultMountPoint | DefaultDriveLetter |
           all | all_local | all_remote | {-F DeviceFileName}}
          [-o MountOptions] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
      or
mmmount Device {MountPoint | DriveLetter}
          [-o MountOptions] [-a | -N {Node[,Node...] | NodeFile | NodeClass}]
```

4.3.3 Unmounting a file system

Some Spectrum Scale administration tasks require you to unmount the file system before they can be performed. You can unmount a Spectrum Scale file system by using the **mmumount** command. If the file system will not unmount, see Chapter 8, “Problem determination” on page 411.

To unmount a Spectrum Scale file system using the **mmumount** command, enter:

```
mmumount device
```

Where *device* is the name of the file system. For example, to unmount the file system ***fs1***, enter:

```
mmumount fs1
```

It is possible to unmount file system *fs1* on all nodes in the Spectrum Scale cluster by using the **-a** flag. Issue this command to unmount:

```
mmumount fs1 -a
```

4.3.4 Creating an IBM Spectrum Scale file system

To add a file system to a cluster, use the **mmcrfs** command. The command has several options as shown in Example 4-30.

Example 4-30 The mmcrfs command options

```
[root@fslinux1 ~]# mmcrfs
mmcrfs: Missing arguments.
Usage:
   mmcrfs Device {"DiskDesc[;DiskDesc...]" | -F StanzaFile}
          [-A {yes | no | automount}] [-B BlockSize] [-D {posix | nfs4}]
          [-E {yes | no}] [-i InodeSize] [-j {cluster | scatter}]
          [-k {posix | nfs4 | all}] [-K {no | whenpossible | always}]
          [-L LogFileSize] [-m DefaultMetadataReplicas]
          [-M MaxMetadataReplicas] [-n NumNodes] [-Q {yes | no}]
          [-r DefaultDataReplicas] [-R MaxDataReplicas]
          [-S {yes | no | relatime}] [-T Mountpoint] [-t DriveLetter]
          [-v {yes | no}] [-z {yes | no}] [--filesetdf | --nofilesetdf]
          [--inode-limit MaxNumInodes[:NumInodesToPreallocate]]
          [--log-replicas LogReplicas] [--metadata-block-size BlockSize]
          [--perfileset-quota | --noperfileset-quota]
          [--mount-priority Priority] [--version VersionString]
          [--write-cache-threshold WriteDataLogThreshold]
```

Parameters and several options for the **mmcrfs** command in Spectrum Scale 4.1:

Device	Name that you want to give to this new device.
DiskDesc	The disk descriptors list, or directly the disk descriptions file (-F).
-F DescFile	The file that has been modified by the **mmcrnsd command**.
-T mnt point	Sets the mount point, here an example of a newly created file system.
-B Block Size	Defines the block size to be used for the file system. Possible values are: 64 K, 128 K, 256 K, 512 K, 1 M, 2 M, 4 M, 8 M, or 16 M.

-M MaxMetadataReplicas

This defines the maximum number of copies of metadata for all files in the file system. The default is 2 and the allowable values are 1, 2, or 3, but it cannot be less than the value of *DefaultMetadataReplicas*. This value cannot be changed.

-m DefaultMetadataReplicas

Defines the default replication factor for metadata blocks. This value must be equal or less than MaxMetadataReplicas.

-R MaxDataReplicas The maximum number of copies of data blocks for a file. The default is 2 and the allowable values are 1, 2, and 3, but it cannot be less than the value of *DefaultDataReplicas*. This value cannot be changed.

-r DefaultDataReplicas

Defines the default number of copies of data blocks for a file. This value must be equal or less than MaxDataReplicas.

The **mmcrfs** command can use as the input file the same file used to create a new NSD, as described in 4.4.2, “Creating NSDs” on page 209, or create a new file based on the NSD name that you are planning to use. In our scenario, the **mmcrfs** command is used to create a new file system using `nsd.file.nsd005` as an input file as shown in Example 4-31.

Example 4-31 The contents of nsd.file.nsd005 input file and the mmcrfs command

```
[root@fslinux1 gpfs]# cat nsd.file.nsd005
 %nsd:
  device=/dev/dm-6
  nsd=NSD005
  servers=fslinux1,fslinux2
  usage=dataAndMetadata
  failureGroup=1
  pool=system

[root@fslinux1 gpfs]# mmcrfs brazilfs -F nsd.file.nsd005 -T /brazil

The following disks of brazilfs will be formatted on node fslinux1:
    NSD005: size 10240 MB
Formatting file system ...
Disks up to size 109 GB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/brazilfs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice the output line:

```
Disks up to size 1.1 TB can be added to storage pool 'system'.
```

This value is calculated during file system creation time when the **mmcrfs** command is issued because on each disk there is a portion where Spectrum Scale stores its control information and a disk bitmap that shows, in the disk, which area is allocated or not. Therefore, if you try

to add a larger disk to the same storage pool, and the space is not enough, Spectrum Scale provides, in advance, the maximum disk size you can add to this storage pool. After creation of a file system, you can use the **mmdf** command to determine what is the minimum disk size in the file system.

A parameter that can affect Spectrum Scale performance is the block size (**-B**). It is suggested that you use the same block size value that is used for the application that writes in your Spectrum Scale file system. For more information about block sizes, see *GPFS V4.1: Concepts, Planning, and Installation Guide*, GA76-0441-00.

4.3.5 Removing a file system

To remove a file system from your cluster, simply use the **mmdelfs** command. However, first be sure that the file system is unmounted in every node in the cluster. Keep in mind that this step does not delete the disks, but the NSD will be seen as "free" after the file system is deleted. If the disks are damaged or not available, the **-p** parameter must be used. Although Spectrum Scale cannot mark them as free, the file system is removed anyway.

Warning: The use of the **-p** flag should be used carefully to avoid data loss.

We deleted the previously created `/dev/brazilfs` file system as shown in Example 4-32.

Example 4-32 Using the mmdelfs command to completely delete a file system

```
[root@fslinux1 gpfs]# mmunmount brazilfs -a
Fri Oct 24 14:47:08 EDT 2014: mmunmount: Unmounting file systems ...
[root@fslinux1 gpfs]# mmdelfs brazilfs
All data on the following disks of brazilfs will be destroyed:
    NSD005
Completed deletion of file system /dev/brazilfs.
mmdelfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root@fslinux1 gpfs]# mmlsnsd
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 GPFS          NSD001       fslinux1,fslinux2
 GPFS          NSD002       fslinux1,fslinux2
 GPFS          NSD003       fslinux1,fslinux2
 remotefs      NSD004       fslinux1,fslinux2
 (free disk)   NSD005       fslinux1,fslinux2
```

4.3.6 Repairing a file system

The **mmfsck** command finds and repairs conditions that can cause problems in your file system. Starting in Spectrum Scale 4.1, a new feature was added to make the file system repair process faster. In previous versions, when a disk has problems and goes to down state, during the period where the disk is down, you might have only a small write in the file system that is residing in the down disk and only a few blocks are changed, but the repair command tried to recover the whole file. Now, in Spectrum Scale 4.1, the **mmfsck** command tries to repair only the modified blocks. This feature also added a new file system attribute, the *--rapid-repair* that is enabled by default. This attribute can be seen in 4.3.8, "Listing file system attributes" on page 205.

The **mmfsck** command operates in two modes: online and offline.

- The *online mode* operates on a mounted file system and is chosen by issuing the -o option and it only checks and recovers deallocated blocks. If a Spectrum Scale file operation fails due to an out-of-space condition, the cause may be disk blocks that have become unavailable after repeated node failures. The corrective action taken is to mark the block free in the allocation map. Any other inconsistencies found are only reported, not repaired.
- The *offline mode* operates on an unmounted file system and, in general, it is unnecessary to run **mmfsck** in offline mode unless under the direction of the IBM Support Center to repair some types of inconsistencies. The offline mode checks for these file inconsistencies that might cause problems:
  - Blocks marked allocated that do not belong to any file. The corrective action is to mark the block free in the allocation map.
  - Files and directories for which an inode is allocated and no directory entry exists, known as *orphaned files*. The corrective action is to create directory entries for these files in a lost+found subdirectory in the root directory of the fileset to which the file or directory belongs. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. The index number of the inode is assigned as the name. If you do not allow the **mmfsck** command to reattach an orphaned file, it asks for permission to delete the file.
  - Directory entries pointing to an inode that is not allocated. The corrective action is to remove the directory entry.
  - Incorrectly formed directory entries. A directory file contains the inode number and the generation number of the file to which it refers. When the generation number in the directory does not match the generation number stored in the file's inode, the corrective action is to remove the directory entry.
  - Incorrect link counts on files and directories. The corrective action is to update them with accurate counts.
  - Policy files that are not valid. The corrective action is to delete the file.
  - Various problems related to file sets: missing or corrupted fileset metadata, inconsistencies in directory structure related to file sets, missing or corrupted fileset root directory, other problems in internal data structures.

You cannot run the **mmfsck** command on a file system that has disks in a down state. You must first run the **mmchdisk** command to change the state of the disks to unrecovered or up. To display the status of the disks in the file system, issue the **mmlsdisk** command.

For example, to check the file system gpfs2 while it is mounted, enter the command as shown in Example 4-33.

Example 4-33 The mmfsck in a mounted file system (-o option)

```
root@tsmsrv:/.ssh>mmfsck  gpfs2 -o
Checking "gpfs2"
   5 % complete on Tue Oct 21 16:15:50 2014
Checking inodes
 100 % complete on Tue Oct 21 16:16:07 2014
13107200 subblocks
                64254   allocated
                    0   unreferenced
                    0   deallocated

                 1754 addresses
```

```
                    0   suspended
File system is clean.
Exit status 0:10:0
```

To perform the **mmfsck** operation in an unmounted file system, run the **mmfsck** command without options as shown in Example 4-34.

Example 4-34 The mmfsck to perform a full check in an unmounted file system

```
root@tsmsrv:/.ssh>mmfsck gpfs2
Checking "gpfs2"
Checking reserved files
   5 % complete on Tue Oct 21 16:18:13 2014
Checking inodes
  45 % complete on Tue Oct 21 16:18:14 2014
Checking inode map file
  50 % complete on Tue Oct 21 16:18:18 2014
  52 % complete on Tue Oct 21 16:18:18 2014
  54 % complete on Tue Oct 21 16:18:18 2014
Checking ACL file records
  56 % complete on Tue Oct 21 16:18:18 2014
Checking directories and files
Checking log files
Checking extended attributes file
Checking allocation summary file
Checking policy file
Checking metadata of filesets
Checking file reference counts
  98 % complete on Tue Oct 21 16:18:20 2014
Checking file system replication status
 100 % complete on Tue Oct 21 16:18:20 2014

               410624 inodes
                   40   allocated
                    0   repairable
                    0   repaired
                    0   damaged
                    0   deallocated
                    0   orphaned
                    0   attached
                    0   corrupt ACL references

             13107200 subblocks
                64254   allocated
                    0   unreferenced
                    0   duplicates
                    0   deletable
                    0   deallocated

                 1754 addresses
                    0   suspended
                    0   duplicates
                    0   reserved file holes found
                    0   reserved file holes repaired
File system is clean.
```

Note: If the **mmfsck** command is issued with the **-y** flag, all inconsistencies are automatically corrected.

4.3.7 Reducing file system fragmentation

Fragmentation of files in a file system cannot be avoided. When Spectrum Scale has to write a file, a free full block has to be allocated and when the file is closed the last logical block of data is reduced to the actual number of subblocks required, creating in this way a fragmented block.

The Spectrum Scale **mmdefragfs** command can reduce file system fragmentation. The **mmdefragfs** command moves pieces of fragmented data to another fragmented block where there is free space ultimately resulting in free full blocks. Use the **mmdefragfs -i** command to query how much your file system is fragmented, helping you decide whether or not to run defragmentation process. The defragmentation process can be run with the file system mounted or unmounted. When it is run on a mounted file system, the process might be slow because of possible data read/write conflicts that occur when your application tries to read or write a block at the same time that the **mmdefragfs** command wants to move it. If the file system is heavily accessed, this situation can happen often. Running time of this command depends on how much data there is in the file system, especially how much data is fragmented, and therefore, how many blocks Spectrum Scale has to relocate.

Example 4-35 shows the **mmdefragfs** command run on the file system. First, the **mmdefragfs** command is run with the **-i** parameter, only to query data. Second, the command is run without parameters, performing the actual defragmentation.

Example 4-35 Querying fragmentation information with mmdefragfs -i

```
root@tsmsrv:/.ssh>mmdefragfs gpfs2 -i
Checking fragmentation in file system 'gpfs2'...
"gpfs2"   410624 inodes:    4038 allocated / 406586 free

                         free subblk    free
disk            disk size  in full  subblk in      %       %
name          in nSubblk   blocks   fragments free blk blk util
--------------- --------- --------- --------- --------  -------
gpfs1nsd          3276800   3260640        90   99.507   99.997
gpfs2nsd          3276800   3260672        82   99.508   99.997
gpfs3nsd          3276800   3260672        59   99.508   99.998
gpfs4nsd          3276800   3260672        59   99.508   99.998
                 --------- --------- ---------           -------
(total)          13107200  13042656       290            99.998
```

When running the **mmdefragfs** command, you have to verify the *blk util* column. Consider the number 100 for file systems that are not fragmented. If your file system has the *blk util* over 50 or less, consider running the **mmdefragfs** command.

To efficiently defragment your file system, it is suggested to unmount it before issuing the **mmdefragfs** command as shown in Example 4-36 on page 205.

Example 4-36 Unmounting and running the mmdefragfs command

```
root@tsmsrv:/.ssh>mmumount gpfs2 -a
Tue Oct 21 17:12:00 EDT 2014: mmumount: Unmounting file systems ...

root@tsmsrv:/.ssh>mmdefragfs gpfs2
Defragmenting file system 'gpfs2'...
 100.00 % complete on Tue Oct 21 17:12:33 2014  ( 410624 inodes with total 1751 MB data
processed)

               free subblk              free
disk            in full               subblk in          %             %
name            blocks        blk     fragments        free blk     blk util
              before    after freed   before    after before after before after
---------- ---------- ---------- ---------- ---------- ------------ ------------
gpfs1nsd    3260640  3260640     0       90       90  99.51 99.51  100.00 100.00
gpfs2nsd    3260672  3260640     0       82      106  99.51 99.51  100.00 100.00
gpfs3nsd    3260672  3260672     0       59       59  99.51 99.51  100.00 100.00
gpfs4nsd    3260672  3260672     0       59       59  99.51 99.51  100.00 100.00
           ---------- ---------- ----------                        ------------
(total)    13042656 13042624     0      290      314               100.00 100.00

root@tsmsrv:/.ssh>mmmount gpfs2 -a
mmTue Oct 21 17:18:53 EDT 2014: mmmount: Mounting file systems ...
```

4.3.8 Listing file system attributes

To display the file system information and attributes, you can use the **mmlsfs** command. Depending on your configuration, additional information that is set by Spectrum Scale may be displayed to help in problem determination when contacting the IBM Support Center.

If you specify no options with the **mmlsfs** command, all file system attributes are listed. Example 4-37 shows all of the attributes for the file system named *gpfs2*.

Example 4-37 The mmlsfs output for gpfs2 file system

```
root@tsmsrv:/.ssh>mmlsfs gpfs2
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 32768                    Minimum fragment size in bytes
 -i                 4096                     Inode size in bytes
 -I                 32768                    Indirect block size in bytes
 -m                 1                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 1                        Default number of data replicas
 -R                 2                        Maximum number of data replicas
 -j                 cluster                  Block allocation type
 -D                 nfs4                     File locking semantics in effect
 -k                 all                      ACL semantics in effect
 -n                 32                      Estimated number of nodes that will mount file
system
 -B                 1048576                  Block size
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota no                       Per-fileset quota enforcement
 --filesetdf        no                       Fileset df enabled?
 -V                 14.10 (4.1.0.4)          File system version
 --create-time      Tue Oct 21 11:44:20 2014 File system creation time
```

```
 -z                 no                       Is DMAPI enabled?
 -L                 4194304                  Logfile size
 -E                 yes                      Exact mtime mount option
 -S                 no                       Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           yes                      Fast external attributes enabled?
 --encryption       no                       Encryption enabled?
 --inode-limit      410624                   Maximum number of inodes
 --log-replicas     0                        Number of log replicas
 --is4KAligned      yes                      is4KAligned?
 --rapid-repair     yes                      rapidRepair enabled?
 --write-cache-threshold 0                   HAWC Threshold (max 65536)
 -P                 system                   Disk storage pools in file system
 -d                 gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd  Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /gpfs2                   Default mount point
 --mount-priority   0                        Mount priority
```

Note: Some of the attributes displayed by the **mmlsfs** command represent default mount options. Since the scope of mount options is an individual node, it is possible to have different values on different nodes. For exact **mtime** (**-E** option) and suppressed **atime** (**-S** option), the information displayed by the **mmlsfs** command represents the current setting on the file system manager node. If these options are changed with the **mmchfs** command, the change might not be reflected until the file system is remounted.

4.3.9 Changing file system attributes

To change file system attributes, use the **mmchfs** command. All files created after issuing the **mmchfs** command take on the new attributes. Existing files are not affected. Several options displayed by the **mmlsfs** command represent an attribute that can be changed by the **mmchfs** command.

For example, to change the default data replication factor to 2 for the file system gpfs2, enter:

```
mmchfs gpfs2 -r 2
```

To display the changes, enter:

```
mmlsfs gpfs2 -r
```

The system displays information similar to:

```
flag value description
------------------- ------------------------ -----------------------------------
-r 2 Default number of data replicas
```

Spectrum Scale 4.1.0.4 now provides the option to change the default **LogFileSize** attribute (**-L** option) that is set during file system creation. The default value is 4 M as shown in Example 4-38.

Example 4-38 The mmlsfs command

```
root@tsmsrv:/gpfs>mmlsfs gpfs1 -L
flag                value                    description
------------------- ------------------------ -----------------------------------
 -L                 4194304                  Logfile size
```

Usually, the default size works for mostly workloads but for some specific applications, a larger value might improve the performance for metadata intensive workloads, such as creating and deleting many small files or performing extensive block allocation and deallocation of large files.

Before the option to change the default **LogFileSize**, the only way to change this attribute was to delete and re-create the file system.

You can change this attribute with the **mmchfs** command. The possible values depend on your file system configuration.

-L LogFileSize

Specifies the size of the internal log file. The default size is 4 MB or 32 times the file system block size, whichever is smaller. The minimum size is 256 KB and the maximum size is 32 times the file system block size or 16 MB, whichever is smaller. Specify this value with the K or M character, for example: 8 M.

To change the default value to 128 M, use the **mmchfs** command as shown in Example 4-39.

Example 4-39 Changing the default log file size

```
root@tsmsrv:/gpfs>mmchfs /dev/gpfs1 -L 128M
mmchfs: Attention: You must restart the GPFS daemons before the new log file
size takes effect. The GPFS daemons can be restarted one node at a time.
When the GPFS daemon is restarted on the last node in the cluster, the new
log size becomes effective.

root@tsmsrv:/gpfs>mmlsfs gpfs1 -L
flag                value                    description
------------------- ------------------------ -----------------------------------
 -L                 134217728                Logfile size
```

As informed after the **mmchfs** command is issued, you must stop and start the Spectrum Scale cluster in order to activate the new *Logfile size.*

However, the **mmlsfs** shows you the new attribute. It will not inform you if the new value is already in use. To ensure that the new *Logfile size* value is already being used, you can use the **mmfsadm** command, as shown in Example 4-40.

Example 4-40 The mmfsadm dump command

```
root@tsmsrv:/gpfs>mmfsadm dump stripe |grep "log files"
    log files: 134184960 data bytes, max inode size 16384
```

Usually the value of 128 M is enough for workloads that require a larger *Logfile size* than the default value. Consult your IBM representative when planning to change the default file system *Logfile size*.

4.3.10 Optimizing extended attributes: The fastea option

GPFS version 3.4 introduced a feature regarding extended attributes for files called *fastea*. With the **fastea** attribute enabled, the file systems can read and write extended attributes more quickly than previous versions because it has a completely new organization regarding where extended attributes data is written. Versions before 3.4, had at the beginning of the file, an indicator that provided information about whether extended attributes were specified for

that file. If extended attributes were set, GPFS had to determine and read where this information was written, losing time in the process. When the fastea is enabled, extended attributes are written directly on file inodes, so the reading of these attributes is set much faster.

If you have file systems created with previous a version of GPFS, use the **mmmigratefs** command to migrate the file system to enable the fastea option.

In Spectrum Scale 4.1, the **mmigratefs** command was updated to include the *--online* and *--offline* options:

```
mmmigratefs Device [--fastea] [--online | --offline]

   --fastea
      Convert the existing extended attributes to the new format required for
      storing the attributes in the file's inode and thereby allowing for faster
      extended-attribute access.

   --online
      Allows the mmmigratefs command to run while the file system is mounted.

   --offline
      Allows the mmmigratefs command to run while the file system is unmounted.
```

To determine whether your file system is using the new faster method, use the command as shown in Example 4-41.

Example 4-41 The mmlsfs command with --fastea option

```
[root@fslinux1 /]# mmlsfs GPFS --fastea
flag               value                    description
------------------ ------------------------ -----------------------------------
 --fastea          Yes                      Fast external attributes enabled?
```

To enable fast extended attribute access for file system *GPFS2*, issue this command:

```
mmmigratefs GPFS2 --fastea
```

The system displays information similar to Example 4-42.

Example 4-42 Enabling the fastea attribute

```
[root@fslinux1 gpfs]#mmmigratefs GPFS2 --fastea
Enabling fastea support
11.19 % complete on Thu Oct 23 13:50:24 2014 ( 167936 inodes 328 MB)
33.21 % complete on Thu Oct 23 13:51:56 2014 ( 498260 inodes 973 MB)
100.00 % complete on Thu Oct 23 13:52:04 2014
Finalizing upgrade
11.19 % complete on Thu Oct 23 13:52:27 2014 ( 167936 inodes 328 MB)
26.78 % complete on Thu Oct 23 13:53:07 2014 ( 401834 inodes 785 MB)
100.00 % complete on Thu Oct 23 13:53:27 2014
Feature 'fastea' is now enabled on "GPFS2".
```

Note: For more file system administration tasks, see *GPFS V4.1: Administration and Programming Reference,* SA23-1452-00 under the "*Managing File Systems"* chapter.

4.4 Managing IBM Spectrum Scale disks

When using disks in a Spectrum Scale cluster, you can have disks directly attached to each node in the cluster, disks managed as network shared disk servers, or a combination of the two. In order to use a disk in a Spectrum Scale cluster, we use NSDs to represent physical disks. These topics cover some of the most common operations regarding disks in a Spectrum Scale cluster, such as:

- Displaying disks
- Creating NSDs
- Adding a disk to a file system
- Deleting disks from a file system

4.4.1 Displaying disks

You can display the disks that belong to your Spectrum Scale cluster by issuing the **mmlsnsd** command.

The default is to display information for all disks defined to the cluster (**-a**). Otherwise, you can choose to display the information for a particular file system (**-f**) or for all disks that do not belong to any file system (**-F**).

To display the default information for all of the NSDs belonging to the cluster, enter the **mmlsnsd** command as shown in Example 4-43.

Example 4-43 The mmlsnsd command

```
[root@fslinux1 gpfs]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 GPFS          NSD001       fslinux1,fslinux2
 GPFS          NSD002       fslinux1,fslinux2
 GPFS          NSD003       fslinux1,fslinux2
 GPFS          NSD004       fslinux1,fslinux2
```

4.4.2 Creating NSDs

When planning to use more disks in a Spectrum Scale cluster, the first step is to create the NSDs on the existing and available disks on the nodes. To create new NSDs, we use the **mmcrnsd** command. This is the first Spectrum Scale step in preparing disks for use by a Spectrum Scale file system. The input to this command consists of a file containing NSD stanzas describing the properties of the disks to be created. This file can be updated as necessary by the **mmcrnsd** command and can be supplied as input to the **mmcrfs**, **mmadddisk**, or **mmrpldisk** commands.

The **mmcrnsd** command must have the following syntax:

```
mmcrnsd -F StanzaFile [-v {yes | no}]
```

Where:

-F StanzaFile

Specifies the file containing the NSD stanzas for the disks to be created. NSD stanzas have this format:

```
device=DiskName
nsd=NsdName
servers=ServerList
usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
failureGroup=FailureGroup
pool=StoragePool
```

The following parameters are used to create the stanza file:

device=DiskName

On UNIX, the block device name appearing in /dev for the disk you want to define as an NSD. Examples of disks that are accessible through a block device are SAN-attached disks. If server nodes are specified, DiskName must be the /dev name for the disk device of the first listed NSD server node.

On Windows operating systems, the disk number (for example, 3) of the disk you want to define as an NSD. Disk numbers appear in Windows Disk Management console and the DISKPART command-line utility. If a server node is specified, DiskName must be the disk number from the first NSD server node defined in the server list.

nsd=NsdName

Specify the name that you want for the NSD to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string 'gpfs'.

servers=ServerList

Is a comma-separated list of NSD server nodes. You may specify up to eight NSD servers in this list. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use the next available server on the list.

usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}

Specifies the type of data to be stored on the disk:

dataAndMetadata

Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly

Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly

Indicates that the disk contains metadata and does not contain data.

descOnly

Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations.

localCache

Indicates that the disk is to be used as a local read-only cache device.

failureGroup=FailureGroup

Identifies the failure group to which the disk belongs. A failure group identifier can be a simple integer or a topology vector consisting of up to three comma-separated integers. The default is -1, which indicates that the disk has no point of failure in common with any other disk.

pool=StoragePool

Specifies the name of the storage pool to which the NSD is assigned. This clause is ignored by the **mmcrnsd** command and is passed unchanged to the output file produced by the **mmcrnsd** command.

-v {yes | no}

Verify that the disks are not already formatted as an NSD. A value of -v yes specifies that the NSDs are to be created only if each disk has not been formatted by a previous invocation of the **mmcrnsd** command, as indicated by the NSD volume ID on sector 2 of the disk. A value of -v no specifies that the disks are to be created irrespective of their previous state. The default is -v yes.

Perform the following steps to add a new disk to a Spectrum Scale file system:

1. Determine whether any already-configured NSDs are available (not yet associated to a file system) by using the **mmlsnd -F** command.
2. To create an NSD, you must know the following information:
   - The name of the device that will be used for NSD creation. You must use the appropriated OS command to identify any unused disk that you plan to add to your new NSD (devices in other nodes in the cluster are automatically discovered).
   - The type of disk usage that this NSD will handle (*dataAndMetadata*, *dataOnly*, *metadataOnly*, and *descOnly* descriptors as described earlier in this chapter).
   - Whether assigning one or more NSD servers to this new NSD is necessary (or whether it is only directly attached to the nodes).
   - The failure group on which to configure it (to have Spectrum Scale manage replica).
   - Whether the new NSD has to be in a storage pool separate from the system.

   Usually, you have to define one or more (up to eight) servers to the NSD if data that is contained in this NSD must also be accessible by nodes that do not have direct access to that disk. When a node must read or write data to a disk on which it does not have direct access, the node has to ask, through the network, for a node that is on the server list for that NSD. Everything is done through the network. If no NSD servers are specified, the disk is seen as a *direct attach* disk. If other servers are configured in the cluster and need to access data on that NSD, they cannot use this server.
3. Certain items, such as the NSD server list, can be changed later with the **mmchnsd** command. Other items, such as the NSD name or the storage pool, are impossible to change. Instead, to change the NSD usage or the failure group, you have to use the **mmchdisk** command.

   In our ITSO environment, we have a cluster with two Red Hat Linux servers using IBM SAN Volume Controller disks. The first step is to list the available disks as shown in Example 4-44.

Example 4-44 Using the multipath command to display available disks

```
[root@fslinux1 dev]# multipath -l |grep dm-
mpathe (3600507680191026c400000000000015b) dm-6 IBM,2145
mpathd (3600507680191026c4000000000000159) dm-4 IBM,2145
mpathc (3600507680191026c4000000000000158) dm-3 IBM,2145
mpathb (3600507680191026c4000000000000157) dm-2 IBM,2145
mpathf (3600507680191026c400000000000015a) dm-5 IBM,2145
```

4. Use the **mmlsnsd -M** command to verify which disks are not being used by the cluster as shown in Example 4-45 on page 212.

Example 4-45 The mmlsnsd command

```
[root@fslinux1 dev]# mmlsnsd -M

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
NSD001        149DAC10543EF1ED   /dev/dm-2      fslinux1                 server node
NSD001        149DAC10543EF1ED   /dev/dm-2      fslinux2                 server node
NSD002        149DAC10543EF1F0   /dev/dm-3      fslinux1                 server node
NSD002        149DAC10543EF1F0   /dev/dm-6      fslinux2                 server node
NSD003        149DAC10543EF1F3   /dev/dm-4      fslinux1                 server node
NSD003        149DAC10543EF1F3   /dev/dm-3      fslinux2                 server node
```

The input file contents used by the **mmcrnsd** in our test environment are shown in Example 4-46.

Example 4-46 Input file /root/gpfs/nsd.file used for the mmcrnsd command

```
[root@fslinux1 gpfs]# cat /root/gpfs/nsd.file
%nsd:
  device=/dev/dm-5
  nsd=NSD004
  servers=fslinux1,fslinux2
  usage=dataAndMetadata
  failureGroup=1
  pool=system
```

The new disk is connected to both servers (fslinux1 and fslinux2). The **mmcrnsd** command is shown in Example 4-47.

Example 4-47 Using the mmcrnsd command to create new disks

```
[root@fslinux1 gpfs]# mmcrnsd -F nsd.file.dsk
mmcrnsd: Processing disk dm-5
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root@fslinux1 gpfs]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 GPFS          NSD001       fslinux1,fslinux2
 GPFS          NSD002       fslinux1,fslinux2
 GPFS          NSD003       fslinux1,fslinux2
 (free disk)   NSD004       fslinux1,fslinux2
```

The new NSD called *NSD004* was created and is available for usage.

4.4.3 Adding a disk to a file system

Adding a disk to a file system can be a regular occurrence if your file systems are rapidly growing in size, and you need more free disk space. Spectrum Scale gives you the option to add disks to your file system online, while the cluster is active, and the new empty space can be used as soon as the command completes. When you add a new disk using the **mmaddisk** command, you can decide to use the **-r** balance flag if you want all existing files in your file

system to use the new free space. Rebalancing is an I/O-intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. You must consider that during the file system rebalancing, Spectrum Scale operations might be slower than usual, so be sure to do a rebalancing when there is not high I/O on the file system. You can also add a new disk without rebalancing, and do a complete rebalancing later, with the **mmrestripefs** command.

Perform the following steps to add a new disk to a Spectrum Scale file system:

1. Determine whether any already-configured NSDs are available (not yet associated to a file system) by using the **mmlsnd -F** command.
2. If there are no available NSDs that are already configured in the cluster, you must create a new NSD on a disk device as shown in 4.4.1, “Displaying disks” on page 209.
3. List any available NSDs to be used by your file system by the **mmlsnsd** command as shown in Example 4-48.

Example 4-48 The mmlsnsd command

```
[root@fslinux1 gpfs]# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 GPFS          NSD001       fslinux1,fslinux2
 GPFS          NSD002       fslinux1,fslinux2
 GPFS          NSD003       fslinux1,fslinux2
 (free disk)   NSD004       fslinux1,fslinux2
```

4. A free NSD disk is available and ready to be added to a file system with the **mmadddisk** command. The **mmadddisk** command can use as input file the same file used to create the new *NSD004*. In our ITSO lab environment, a file system named Spectrum Scale receives a new disk as shown in Example 4-49.

Example 4-49 mmadddisk to add NSD in the file system

```
[root@fslinux1 gpfs]# mmadddisk GPFS -F  nsd.file.1

The following disks of GPFS will be formatted on node fslinux1:
    NSD004: size 10240 MB
Extending Allocation Map
Checking Allocation Map for storage pool system
Completed adding disks to file system GPFS.
mmadddisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The new disk is formatted and added to the existing file system, named **GPFS**, as shown in Example 4-50.

Example 4-50 New disks in use on the file system

```
[root@fslinux1 gpfs]# mmdf GPFS
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 94 GB)
NSD001               10485760        1 Yes      Yes        10287104 ( 98%)           600 ( 0%)
NSD002               10485760        1 Yes      Yes        10285824 ( 98%)           376 ( 0%)
NSD003               10485760        1 Yes      Yes        10284800 ( 98%)           376 ( 0%)
```

```
NSD004              10485760        1 Yes      Yes        10419712 ( 99%)           376 ( 0%)
             -------------                                -------------------- -------------------
(pool total)        41943040                                41277440 ( 98%)          1728 ( 0%)

             =============                                ==================== ===================
(total)             41943040                                41277440 ( 98%)          1728 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:           61818
Number of allocated inodes:      65856
Maximum number of inodes:        65856
```

4.4.4 Deleting disks from a file system

Usually this operation is used when there is a plan to migrate data from old storage to a new storage subsystem. Before deleting a disk, use the **mmdf** command to determine whether there is enough free space on the remaining disks to store the file system data

When deleting disks from a file system, the disks may or may not be available. If the disks being deleted are still available, Spectrum Scale moves all of the data from those disks to the disks remaining in the file system.

However, if the disks being deleted are damaged, either partially or permanently, it is not possible to move all of the data and you receive I/O errors during the deletion process and the process fails.

Specify the file system and the names of one or more disks to delete with the **mmdeldisk** command. For example, to delete the disk NSD004 from the file system *GPFS*, enter the **mmdeldisk** command as shown in Example 4-51.

Example 4-51 Deleting disk NSD004 from file system GPFS

```
[root@fslinux1 gpfs]# mmdeldisk GPFS NSD004
Deleting disks ...
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Tue Oct 21 18:40:28 2014  (     65856 inodes with total
391 MB data processed)
Scan completed successfully.
Checking Allocation Map for storage pool system
tsdeldisk completed.
mmdeldisk: Propagating the cluster configuration data to all affected nodes.  This
is an asynchronous process.
```

Note: Deleting disks can cause some performance degradation while the **mmdeldisk** is running. To avoid any performance issues, it is suggested to plan the execution of this task when the cluster is not running under intensive workload.

4.4.5 Replacing disks in an IBM Spectrum Scale file system

Replacing an existing disk in a Spectrum Scale file system with a new one is the same as performing a delete disk operation followed by an add disk.

When replacing disks in a Spectrum Scale file system, first decide if you will:

1. Create new disks by using the **mmcrnsd** command.
2. Select NSDs no longer in use by another Spectrum Scale file system. Issue the **mmlsnsd -F** command to display the available disks.

The **mmrpldisk** command needs the following inputs:

```
mmrpldisk Device DiskName
```

Where:

Device Is the file system name

DiskName Is the NSD name where the first entry will be the source and the second entry will be the destination disk

To replace a disk in the file system, use the **mmrpldisk** command. For example, to replace the NSD named *NSD003* in file system *GPFS* with the existing NSD named *NSD004*, which is no longer in use by another file system, enter:

```
mmrpldisk GPFS NSD003 NSD004
```

The output should be similar to Example 4-52.

Example 4-52 Using mmrpldisk to replace NSD003 for NSD004

```
[root@fslinux1 gpfs]# mmrpldisk GPFS NSD003 NSD004
Replacing NSD003 ...

The following disks of GPFS will be formatted on node fslinux1:
    NSD004: size 10240 MB
Extending Allocation Map
Checking Allocation Map for storage pool system
Completed adding disks to file system GPFS.
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Wed Oct 22 16:05:38 2014  (     65856 inodes with total
391 MB data processed)
Scan completed successfully.
Checking Allocation Map for storage pool system
```

```
Done
mmrpldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note: For more disk administration commands, see *GPFS V4.1: Administration and Programming Reference,* SA23-1452-00 under the "*Managing disks"* chapter.

4.5 Managing IBM Spectrum Scale data migration

Spectrum Scale enables you to migrate data from old storage to new storage while the system is running. You can use the default Spectrum Scale features to automatically move data within the same file system and between every NSD that has been configured or there is a new feature called *AFM-based NFS migration* to help system administrators in migrating data from any earlier storage appliance to a Spectrum Scale cluster, via NFS protocol. In this section, we show the data migration options.

4.5.1 Migrating data using NSDs in GPFS file systems

Using NSDs to migrate data can be accomplished in simple steps. Basically there are two main Spectrum Scale commands that are used during this operation:

- The **mmadddisk** command to add the NSD created on new storage to the file system.
- The **mmdeldisk** command to remove disks belonging to the old storage and have data migrated to the new storage.

When a new disk is assigned to an NSD, a user storage pool can be created. User storage pools are used to group disks with similar properties to have benefit in reliability and price per performance. After the addition of new disks to the file system, you can delete the old disks. During this operation, the data that resides on old disks is moved to new disks, so there is the need to have sufficient free space in the file system to accommodate all the data. If the deleted disk is the last disk available in a user-defined storage pool, also the user created storage pool is deleted.

When planning this migration, ensure that the new disk has the same or larger capacity than the current disk, otherwise the migration fails.

For example, in order to migrate your Spectrum Scale data from your old storage to the new equipment, using NSDs and Spectrum Scale file system, the following steps can be followed:

1. Configure the new storage, so you can correctly see new disks in your OS.
2. Make sure that the new disks are available and ready to use. Example 4-53 shows the AIX **lspv** command usage to display the available disks.

Example 4-53 Listing disks

```
root@tsmsrv:/>lspv
hdisk0          00f6f5d0d1e72b9c                    rootvg          active
hdisk1          00f6f5d01f23b2bb                    tsmdbvg         active
hdisk2          00f6f5d01f2a2880                    tsmdbvg         active
hdisk3          00f6f5d01fc3fabb                    tsmdisk1
hdisk4          none                                tsmdisk2
hdisk5          none                                tsmdisk3
hdisk6          none                                tsmdisk4
```

```
hdisk7          none                                    tsmclient1
hdisk8          00f6f5d020c904d8                        None
hdisk9          00f6f5d020d6177d                        tsmdbvg         active
```

3. Create an NSD for the new disk to use as shown in Example 4-54.

   The `migrationdisk.txt` was the input file for the NSD creation, which has the following contents:

```
%nsd:
device=hdisk8
usage=dataAndMetadata
nsd=migratensd
```

Example 4-54 Creating a new NSD using the available disk

```
root@tsmsrv:/gpfs>mmcrnsd -F migrationdisk.txt -v yes
mmcrnsd: Processing disk hdisk8
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

4. Make sure that the NSD was created correctly. Use the **mmlsnsd** command to show the NSDs as shown in Example 4-55.

Example 4-55 The mmlsnsd command shows the new NSD called migratensd

```
root@tsmsrv:/gpfs>mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 tsmclient     tsmclient1   (directly attached)
 tsmgpfs       tsmdisk1     (directly attached)
 tsmgpfs       tsmdisk2     (directly attached)
 tsmgpfs       tsmdisk3     (directly attached)
 tsmgpfs       tsmdisk4     (directly attached)
 (free disk)   migratensd   (directly attached)
```

5. In Example 4-56, the tsmclient Spectrum Scale file system has only one NSD called *tsmclient1*.

Example 4-56 The mmdf command to show the size of the Spectrum Scale file system

```
root@tsmsrv:/tsmclient>mmdf tsmclient
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 192 GB)
tsmclient1           10485760       -1 yes      yes         4218880 ( 40%)         22016 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         10485760                               4218880 ( 40%)         22016 ( 0%)

                =============                         ==================== ===================
(total)              10485760                               4218880 ( 40%)         22016 ( 0%)
Inode Information
-----------------
Number of used inodes:            6818
Number of free inodes:           59230
Number of allocated inodes:      66048
```

```
Maximum number of inodes:        66048
```

6. To start the data migration now, add the newly created NSD to the existing file system by using the **mmadddisk** command as shown in Example 4-57.

   When running the **mmadddisk** command, you can use the same input file used to create the NSDs. In this example, it is `migrationdisk.txt`.

Example 4-57 Adding disks to an existing Spectrum Scale file system

```
root@tsmsrv:/gpfs>mmadddisk tsmclient -F migrationdisk.txt
The following disks of tsmclient will be formatted on node tsmcl1:
    migratensd: size 10240 MB
Extending Allocation Map
Checking Allocation Map for storage pool system
Completed adding disks to file system tsmclient.
mmadddisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

7. Now verify if the Spectrum Scale file system has both NSDs (old and new) by using the **mmdf** command as shown in Example 4-58.

Example 4-58 The mmdf command to display Spectrum Scale file system contents

```
root@tsmsrv:/gpfs>mmdf tsmclient
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 192 GB)
tsmclient1           10485760       -1 yes      yes          4218880 ( 40%)         22016 ( 0%)
migratensd           10485760       -1 yes      yes         10419200 ( 99%)           880 ( 0%)
             -------------                         -------------------- -------------------
(pool total)         20971520                           14638080 ( 70%)         22896 ( 0%)

             =============                         ==================== ===================
(total)              20971520                           14638080 ( 70%)         22896 ( 0%)
Inode Information
-----------------
Number of used inodes:            6818
Number of free inodes:           59230
Number of allocated inodes:      66048
Maximum number of inodes:        66048
```

8. After checking that the new NSD was added to the existing Spectrum Scale file system, you can remove the old NSD from the Spectrum Scale file system. Spectrum Scale automatically migrates data to the new NSD. In Example 4-59, the *tsmclient1* NSD is removed.

Example 4-59 Using the mmdeldisk command to remove an old NSD

```
root@tsmsrv:/gpfs>mmdeldisk tsmclient tsmclient1
Deleting disks ...
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
```

```
Scanning user file metadata ...
  22.97 % complete on Tue Oct 21 11:11:29 2014  (     62032 inodes with total       1418 MB data processed)
  27.32 % complete on Tue Oct 21 11:12:16 2014  (     62035 inodes with total       1686 MB data processed)
  49.32 % complete on Tue Oct 21 11:12:40 2014  (     62127 inodes with total       3044 MB data processed)
  56.92 % complete on Tue Oct 21 11:13:02 2014  (     62149 inodes with total       3513 MB data processed)
 100.00 % complete on Tue Oct 21 11:13:13 2014  (     66048 inodes with total       3534 MB data processed)
Scan completed successfully.
Checking Allocation Map for storage pool system
Attention: A disk being removed reduces the number of failure groups to 1,
           which is below the number required for replication: 2.
           New blocks will be allocated from the remaining disks,
           but files will be unreplicated and hence at risk.
tsdeldisk completed.
mmdeldisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

9. After deleting the NSD from the Spectrum Scale file system, you can delete the NSD from the Spectrum Scale configuration. You can run the **mmlsnsd** command to check that the NSD is now a "*free disk*" as shown in Example 4-60.

Example 4-60 Using the mmlsnsd to show the NSD status

```
root@tsmsrv:/gpfs>mmlsnsd
File system   Disk name    NSD servers
---------------------------------------------------------------------------
 tsmclient     migratensd   (directly attached)
 tsmgpfs       tsmdisk1     (directly attached)
 tsmgpfs       tsmdisk2     (directly attached)
 tsmgpfs       tsmdisk3     (directly attached)
 tsmgpfs       tsmdisk4     (directly attached)
 (free disk)   tsmclient1   (directly attached)
```

Now you can delete the NSD from the Spectrum Scale configuration by using the **mmdelnsd** command as shown in Example 4-61.

Example 4-61 Using the mmdelnsd to delete the old NSD

```
root@tsmsrv:/gpfs>mmdelnsd tsmclient1
mmdelnsd: Processing disk tsmclient1
mmdelnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

10. After deleting the NSD from the Spectrum Scale configuration, you can use operating system commands to remove the old disk.
11. After performing data migration by deleting and adding new disks, it is suggested that you rebalance the data across all disks by issuing the **mmrestripefs -r** command. This command will try to move the data for performance optimization.

4.5.2 Migrating data using AFM-based NFS Migration

NFS Migration is a process of migrating data from any earlier storage appliance to a Spectrum Scale cluster, via NFS protocol. The existing storage appliance can be disconnected once the migration is complete. If an IP switch-over is later executed, the migration will be seamless and users or applications will not be affected. AFM NFS Migration can be very useful when upgrading hardware or buying a new system where the data from old hardware needs to be moved to new hardware.

Following are the key goals of data migration:

- Minimize downtime for applications
- Move actual file data along with any permissions, ACLs, and so on, associated with files
- Potentially consolidate data from multiple earlier systems into one more powerful system
- Keep configuration data associated intact if appropriate

When using this migration method, the data migration can be done in the following ways:

- Volume/Block/Disk level migration
- File system migration
- Backup/Restore

AFM provides different options to do planned migration via a **`prefetch`** command or dynamically (by pulling data on demand based on file access), mitigating total downtime required for migration. In the case of dynamic migration, the data is moved slowly based on access over a period of time (ranging from days to weeks or months, based on network speed and other factors). This is called *progressive migration*. Once the data is completely moved, the source of migration can be completely removed. Also, planned and dynamic migration can be mixed based on application or administration requirements.

Since this is a new feature, in order to use migration, the target or new hardware should be running Spectrum Scale 4.1 or later. The data source should be an NFS v3 export and can be either a Spectrum Scale source or a non-Spectrum Scale source. If the source is running a version of GPFS earlier than GPFS 3.4, it is equivalent to a non GPFS data source.

- **GPFS data source**, the AFM moves all Spectrum Scale extended attributes and ACLs, and file sparseness is maintained. It does not migrate file system–specific parameters, like quotas, snapshots, file system level tuning parameters, policies, fileset definitions, encryption keys, and dmapi parameters. AFM migrates data as root, bypassing permission checks. Pre-allocate files at home source are not maintained; that is, only actual data blocks are migrated based on file size. UID namespace between source and target must be maintained.
- **Non GPFS data source**, POSIX permissions, or ACLs are pulled. AFM will not migrate NFS V4/CIFS ACLs, non-posix file system attributes, and sparseness.

AFM migrates data as root, bypassing permission checks. Pre-allocated files at home source are not maintained; that is, only actual data blocks are migrated based on file size. UID namespace between source and target must be maintained.

Before starting migration, the administrator should gather the requirements to choose the correct method to migrate. If home is running GPFS 3.5 or later, run the **`mmafmconfig --enable`** command at home before starting migration. afmAutoEviction must be disabled at fileset creation time. The data source NFS export should be configured as read-only, as AFM will not push any data to the source system. During creation of an AFM fileset, it is advisable to preallocate inodes equal to or greater than the inodes at source, as that many files are known to exist. The maximum inodes should also be set accordingly.

Following are the migration steps for a generic example where the source is an earlier storage device and the target is a Spectrum Scale cluster with AFM:

- On the home cluster (earlier NAS storage), the NFS shares whose data must be migrated are identified.
- All identified shares are exported via NFS.
- At the cache cluster (Spectrum Scale AFM), an AFM fileset is created for each NFS share exported from the home cluster.
- The mode of the file sets is LU or RO, depending upon the type of access required.

- All applications can be moved from pointing to the data source to the cache cluster (Spectrum Scale cluster).
- While users access applications, AFM prefetch tasks can be initiated in parallel on the Spectrum Scale cluster to pull data from the source.
- After all data is brought into the Spectrum Scale AFM cluster, AFM can be disabled. This completes the data migration process.

More information about the AFM and AFM-based NFS migrations can be found under Chapter 6, “Active File Management” on page 291.

4.6 Managing the IBM Spectrum Scale network

In a cluster environment, the Spectrum Scale daemon depends on the correct operation of TCP/IP. These dependencies exist because:

- The communication path between nodes must be built at the first attempt to communicate.
- Each node in the cluster is required to communicate with the cluster manager and the file system manager during start and mount processing.
- Once a connection is established, it must remain active until the Spectrum Scale daemon is shut down on the nodes.

Spectrum Scale might not work properly if there is a firewall that is enabled on the nodes within the cluster. To ensure proper operation, you must either configure the firewall to allow the appropriate ports or disable the firewall in the Spectrum Scale network.

The port number used by the main Spectrum Scale daemon (mmfsd) is controlled with the **tscTcpPort** configuration parameter. The default TCP port number is **1191**.

You can specify a different port number using the **mmchconfig** command:

```
mmchconfig tscTcpPort=PortNumber
```

If your Spectrum Scale cluster is not CCR-enabled, it is permitted to set up different networks for different usage. The network identification occurs during the Spectrum Scale node setup and basically consists on the following types:

- The *administrator network* is the network used by Spectrum Scale administration commands when communicating between nodes. If the **DEFAULT** keyword is specified, the admin interface for the node is set to be equal to the daemon interface for the node.
- The daemon network is used for all non-admin traffic between all nodes.

By having two networks, you can have a slower network for passing administration-related data, and a faster network for passing customer data.

Both the administrator and the daemon interfaces on the nodes can be modified by using the **mmchconfig** command.

For example, in our test cluster, we are using the same interface for both administrator and daemon networks. See Example 4-62 on page 222.

Example 4-62 Our cluster network configuration

```
Node  Daemon node name            IP address       Admin node name             Designation
-----------------------------------------------------------------------------------------
   1   usa                         192.168.101.132  usa                         quorum
   2   germany                     192.168.101.142  germany                     quorum
   4   slovenia                    192.168.101.133  slovenia                    quorum
```

Now we want to switch to the daemon network because we purchased a faster network device and we want to separate the two networks. The network for administration remains on 192.168.101; the new daemon network is now on the 10.10.10 subnet.

The steps are as follows:

1. First, the network must be configured. Therefore, new IP addresses must be correctly configured in the operating system and with resolved names.

   In our lab, we add all the nodes as new entries in the `/etc/hosts` file:

   ```
   10.10.10.1 germany_daemon
   10.10.10.2 usa_daemon
   10.10.10.3 slovenia_daemon
   ```

   These are the IP labels for the network. We configured the interfaces on new adapters and they are working correctly.

2. From Spectrum Scale, configure the new network.

   Before this change, Spectrum Scale must be stopped on all nodes. If it is active somewhere the command tells you. So, **`mmshutdown -a`** command.

   When Spectrum Scale is down, use the **`mmchnode`** command to change Spectrum Scale network interfaces, as shown in Example 4-63.

Example 4-63 The mmchnode command: Changing Spectrum Scale network interfaces

```
usa:/#mmchnode -N usa --daemon-interface=usa_daemon
Tue Oct 21 16:32:03 EDT 2014: mmchnode: Processing node usa
Verifying GPFS is stopped on all nodes ...

mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

usa:/#mmchnode -N germany --daemon-interface=germany_daemon
Tue Oct 21 16:32:55 EDT 2014: mmchnode: Processing node germany
Verifying GPFS is stopped on all nodes ...

mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

usa:/#mmchnode -N slovenia --daemon-interface=slovenia_daemon
Tue Oct 21 16:33:40 EDT 2014: mmchnode: Processing node slovenia
Verifying GPFS is stopped on all nodes ...

mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Now the Spectrum Scale network configuration looks similar to Example 4-64.

Example 4-64 New Spectrum Scale network configuration

```
Node  Daemon node namhe            IP address       Admin node name             Designation
-----------------------------------------------------------------------------------------------
   1   usa_daemon                   10.10.10.2       usa                         quorum
   2   germany_daemon               10.10.10.1       germany                     quorum
   4   slovenia_daemon              10.10.10.3       slovenia                    quorum
```

In this way, Spectrum Scale administration commands use the 192.168.100 network, and Spectrum Scale data uses the 10.10.10 (faster) network.

You can also change the administrator network by using the **`mmchconfig`** command, as always, but instead of using the `--daemon-interface` option, you must use the `--admin-interface` option.

Be sure to pay attention to rsh and SSH configuration; the new interface must be trusted before configuring it in the cluster.

Note: If CCR is enabled, the *--daemon-interface* option is not available so it is not possible to perform the network changes explored in this section.

4.7 Managing IBM Spectrum Scale remote cluster

Spectrum Scale offers the possibility to add a remote cluster to your local Spectrum Scale cluster and mount file systems that are owned and managed by another Spectrum Scale cluster. The procedure to set up remote file system access involves the generation and exchange of authorization keys between the two clusters. In addition, the administrator of the Spectrum Scale cluster that owns the file system needs to authorize the remote clusters that are to access it. Meanwhile, the administrator of the Spectrum Scale cluster that seeks access to a remote file system needs to define to Spectrum Scale the remote cluster and file system whose access is wanted.

4.7.1 Planning and preparation

Each node in the Spectrum Scale cluster requiring access to another cluster's file system must be able to open a TCP/IP connection to every node in the other cluster. Nodes in two separate remote clusters mounting the same file system are not required to be able to open a TCP/IP connection to each other. For example, if a node in *clusterA* mounts a file system from *clusterB*, and a node in *clusterC* wants to mount the same file system, nodes in *clusterA* and *clusterC* do not have to communicate with each other. However, you have to make sure that the interface IP address for the Spectrum Scale daemon communication can be routed between all remote clusters.

If you are planning to use multicluster mounts between GPFS 3.5 and Spectrum Scale 4.1 you need to set *nistCompliance=off* on your Spectrum Scale 4.1 cluster. Spectrum Scale 4.1 provides new authentication ciphers that are not compatible with 3.5.

If you create the remote cluster relationships before turning off NIST compliance you might need to remove the remote cluster definitions before turning off NIST compliance in the Spectrum Scale 4.1 cluster. If you see an error similar to the following, delete the definitions and run it again.

```
mmchconfig nistCompliance=off
mmchconfig: The authentication files associated with cluster cluster2.ibm.com are
not NIST SP800-131A compliant.
mmchconfig: The authentication files associated with cluster cluster2.ibm.com are
not NIST SP800-131A compliant.
mmchconfig: 6027-1639 Command failed. Examine previous error messages to determine
cause.
```

When AFM Gateway nodes are defined in a Spectrum Scale 4.1 cluster, whose file system is mounted by a 3.5. cluster, you need to set *afmHashVersion=1* in the Spectrum Scale 4.1 cluster.

When planning to have more than two clusters that are connected, see "*Accessing GPFS file systems from other GPFS clusters"* in *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00.

4.7.2 Adding a remote cluster

Adding a remote cluster can be performed in three steps:

1. Authorize the remote cluster by using the **mmauth** command.
2. Add the remote cluster by using the **mmremotecluster add** command.
3. Add the remote file system by using the **mmremotefs add** command.

Before starting the procedure (if running Spectrum Scale 4.1), make sure the *gpfs.gskit* is installed on all nodes in the involved clusters before using these instructions.

In our ITSO lab environment, we have the local cluster running on AIX nodes and the remote cluster on Linux nodes. To better understand the local and remote cluster naming, we created Table 4-3. Observe that both *local cluster* and *remote cluster* are terms that are used for testing purposes to illustrate the setup and are not related to Spectrum Scale usage.

Table 4-3 ITSO Lab environment

Cluster name convention	Spectrum Scale cluster name	Spectrum Scale file systems
Local cluster	tsmgpfs.tsmsrv	gpfs2, tsmclient, tsmgpfs
Remote cluster	GPFS.Spectrum	GPFS, remotefs

Example 4-65 shows the local AIX cluster status and the file systems that are configured to this local cluster.

Example 4-65 Displaying local cluster information

```
rroot@tsmsrv:/>mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         tsmgpfs.tsmsrv
  GPFS cluster id:           12742445374757761618
  GPFS UID domain:           tsmgpfs.tsmsrv
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
```

```
-------------------------------------------------------------------------
   1   tsmsrv           172.16.20.153  tsmsrv          quorum-manager
   2   tsmcl1           172.16.20.154  tsmcl1          quorum-manager
   3   tsmcl2           172.16.20.155  tsmcl2
   4   fsaix4           172.16.20.156  fsaix4

root@tsmsrv:/>mmlsmount all
File system gpfs2 is mounted on 4 nodes.
File system tsmclient is mounted on 4 nodes.
File system tsmgpfs is mounted on 4 nodes.
```

Example 4-66 shows the remote Linux cluster, which exports the `remotefs` file system.

Example 4-66 Remote cluster configuration

```
[root@fslinux2 ~]# mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS.Spectrum
  GPFS cluster id:           2497146759570973977
  GPFS UID domain:           STORAGE
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name             IP address     Admin node name  Designation
-------------------------------------------------------------------------------
   1   fslinux1                     172.16.20.157  fslinux1         quorum-manager
   2   fslinux2                     172.16.20.158  fslinux2         quorum-manager

[root@fslinux2 ~]# mmlsmount all
File system GPFS is mounted on 2 nodes.
File system remotefs is mounted on 2 nodes.
```

Perform the following steps:

1. Generate new keys on the local cluster by using the **mmauth** command. It generates a public/private key pair, and places the files in `/var/mmfs/ssl`, as shown in Example 4-67.

Example 4-67 Generate new ssh-keys on cluster1

```
root@tsmsrv:/>hostname
tsmsrv
root@tsmsrv:/>mmauth genkey new
mmauth: Propagating the cluster configuration data to all affected nodes.
mmauth: Command successfully completed
```

2. On the local cluster, you must enable authorization as shown in Example 4-68 on page 226. This example uses AUTHONLY as the authorization setting. When you specify AUTHONLY for authentication, Spectrum Scale checks network connection authorization. However, data sent over the connection is not protected. The *cluster must be down* to perform this step.

Example 4-68 Enabling ssl authorization on local cluster

```
root@tsmsrv:/var/mmfs/ssl>mmauth update . -l AUTHONLY
Verifying GPFS is stopped on all nodes ...
mmauth: Propagating the cluster configuration data to all affected nodes.
mmauth: Command successfully completed
```

3. Now in the local cluster, you can transfer the `/var/mmfs/ssl/id_rsa.pub` file to the remote cluster. It might be useful to rename the key to a distinguished name. Once you configure the multiple remote cluster relationships, where you have to share and distribute your keys within the remote clusters, it is common practice to use dedicated names such as cluster or site names. In this example, a copy that uses the local cluster name was created to identify the file before sending it to the remote cluster as shown in Example 4-69.

Example 4-69 Renaming and transferring public key

```
root@tsmsrv:/var/mmfs/ssl>cp id_rsa.pub id_rsa_tsmgpfs.tsmsrv.pub

root@tsmsrv:/var/mmfs/ssl>id_rsa_tsmgpfs.tsmsrv.pub fslinux1:/var/mmfs/ssl
root@fslinux1's password:
id_rsa_tsmgpfs.tsmsrv.pub                          100% 1792     1.8KB/s   00:00
```

4. Generate new keys on the remote cluster as shown in Example 4-70. The key pair is also placed in `/var/mmfs/ssl`.

Example 4-70 Generate new ssh-keys on remote cluster

```
[root@fslinux1 ssl]# mmauth genkey new
mmauth: Command successfully completed
```

5. Enable the authorization by issuing the **mmauth update** command as shown in Example 4-71.

Example 4-71 Enabling ssl authorization on remote cluster

```
[root@fslinux1 ssl]# mmauth update  . -l AUTHONLY
Verifying GPFS is stopped on all nodes ...
mmauth: Command successfully completed
```

6. Now in the remote cluster, you can transfer the `/var/mmfs/ssl/id_rsa.pub` file to the local cluster. In Example 4-72, a copy was created before sending the file to the local cluster.

Example 4-72 Sending the remote id_rsa.pub to the local cluster

```
[root@fslinux1 ssl]# cp id_rsa.pub id_rsa_GPFS.Spectrum.pub

[root@fslinux1 ssl]# scp id_rsa_GPFS.Spectrum.pub tsmsrv:/var/mmfs/ssl
root@tsmsrv's password:
id_rsa_GPFS.Spectrum.pub                           100% 1785     1.7KB/s   00:00
```

7. On the remote cluster, add the authorization for local cluster to mount file systems owned by the remote cluster as shown in Example 4-73.

Example 4-73 The mmauth add command: Adding authorization for tsmgpfs.tsmsrv cluster

```
[root@fslinux1 ssl]# mmauth add tsmgpfs.tsmsrv -k id_rsa_tsmgpfs.tsmsrv.pub
mmauth: Command successfully completed
```

The following options are used in Example 4-73 on page 226:

- **tsmgpfs.tsmsrv:** Is the real name of the cluster that accesses the file systems.
- **id_rsa_tsmgpfs.tsmsrv.pub:** Is the name of the file transferred as shown in Example 4-69 on page 226.

8. Now you can specify the file system that can be mounted by the local cluster. In this example is the `remotefs` file system as shown in Example 4-74.

Example 4-74 The mmauth grant command: Granting file system authorization

```
[root@fslinux1 ssl]# mmauth grant tsmgpfs.tsmsrv -f /dev/remotefs

mmauth: Granting cluster tsmgpfs.tsmsrv access to file system remotefs:
        access type rw; root credentials will not be remapped.

mmauth: Command successfully completed
```

On the local cluster, you now must add the remote cluster name, node names, and the public key for the remote cluster as shown in Example 4-75.

Example 4-75 Adding remote cluster with mmremotecluster command

```
root@tsmsrv:/var/mmfs/ssl>mmremotecluster add GPFS.Spectrum -n fslinux1,fslinux2 -k
id_rsa_GPFS.Spectrum.pub

mmremotecluster: Command successfully completed
mmremotecluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The following options are used in Example 4-75:

- **GPFS.Spectrum:** Is the real name of the remote cluster.
- **fslinux1, fslinux2:** The node names of the remote cluster.
- **id_rsa_GPFS.Spectrum.pub:** Is the name of the file transferred as shown in Example 4-72 on page 226.

9. You must now identify the file system in the remote cluster that is to be accessed by the local cluster nodes as shown in Example 4-76.

Example 4-76 Using mmremotefs to identify remote file systems to be mounted

```
root@tsmsrv:/>mmremotefs add /dev/local_remotefs -f /dev/remotefs -C GPFS.Spectrum -T /remotefs_loc
mmremotefs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The **mmremotefs** command required the following input information:

- **/dev/local_remotefs**

  Is the device name under which the file system is known in your local cluster.

- **/dev/remotefs**

  Is the real device name for the file system in your remote cluster.

- **GPFS.Spectrum**

  This is the real name of your remote cluster.

- **/remotefs_loc**

  This is your local mount point for the remote file system.

10. Now you can mount the *remotefs* in your local cluster, as shown in Example 4-77.

Example 4-77 Using the mmount command to mount remote file system

```
root@tsmsrv:/.ssh>mmmount /remotefs_loc
Thu Oct 23 16:05:44 EDT 2014: mmmount: Mounting file systems ...
```

11. Verify the mounted file system with the **mmlsmount** command as shown in Example 4-78.

Example 4-78 The mmlsmount command

```
root@tsmsrv:/.ssh>mmlsmount local_remotefs -L
File system local_remotefs (GPFS.Spectrum:remotefs) is mounted on 3 nodes:
  172.16.20.157   fslinux1                  GPFS.Spectrum
  172.16.20.158   fslinux2                  GPFS.Spectrum
  172.16.20.153   tsmsrv                    tsmgpfs.tsmsrv
```

Verify that the file system has the node names and the cluster name of the respective nodes.

Table 4-4 shows the sequence of the operations and commands that are used in this section.

Table 4-4 Summary of commands and tasks during the remote cluster setup

Local cluster - tsmgpfs.tsmsrv	Remote cluster - GPFS.Spectrum
mmauth genkey new	mmauth genkey new
mmauth update . -l AUTHONLY	mmauth update . -l AUTHONLY
Transfer local public key to remote node /var/mmfs/ssl/id_rsa_tsmgpfs.tsmsrv.pub	Transfer local public key to remote node /var/mmfs/ssl/id_rsa_GPFS.Spectrum.pub
mmremotecluster add GPFS.Spectrum -n fslinux1,fslinux2 -k id_rsa_GPFS.Spectrum.pub	mmauth add tsmgpfs.tsmsrv -k id_rsa_tsmgpfs.tsmsrv.pub
mremotefs add /dev/local_remotefs -f /dev/remotefs -C GPFS.Spectrum -T /remotefs_loc	mmauth grant tsmgpfs.tsmsrv -f /dev/remotefs
mmmount /remotefs_loc	

4.7.3 Enabling GPFS replication

GPFS provides multiple features to increase data availability of your system. One feature is the file system synchronous replication. You can set the GPFS replication to work on all the file systems or only on some files. Currently, the maximum number of copies allowed by GPFS is three. This means that three copies of data and metadata are kept replicated to ensure data availability. You can choose to have only metadata replicated to save disk space and to increase performance because enabled replicas require additional write times. GPFS replica is based on failure group configuration. When you add new disks to the GPFS file system, you can configure to which failure group the disks belong. You have to configure the same failure group for disks that have in common the same point of failure, for example, the same disk storage system. If you have one storage system in one location and one storage system in another location, you configure the disks belonging to separate storage systems in separate failure groups. In this way, GPFS knows that it has to replicate data and metadata from one storage system to the other. In a complete storage system failure, GPFS has all the data correctly replicated on the other storage environment.

This section describes the following replica characteristics:

- Default metadata
- Maximum metadata
- Default data
- Maximum data

Additional information about the GPFS replication parameters is in the *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00.

Default metadata replicas

The default number of copies of metadata for all files in the file system may be specified at file system creation by using the **-m** option on the **mmcrfs** command, or changed later by using the **-m** option on the **mmchfs** command. This value must be equal to or less than *MaxMetadataReplicas*, and cannot exceed the number of failure groups with disks that can store metadata. The allowable values are 1, 2, or 3 with a default of 1.

Maximum metadata replicas

The maximum number of copies of metadata for all files in the file system can be specified at file system creation by using the **-M** option on the **mmcrfs** command. The default is 2. The allowable values are 1, 2, or 3 but it cannot be lower than the value of *DefaultMetadataReplicas*. This value cannot be changed.

Default data replicas

The default replication factor for data blocks can be specified at file system creation by using the **-r** option on the **mmcrfs** command, or changed later by using the **-r** option on the **mmchfs** command. This value must be equal to or less than *MaxDataReplicas*, and the value cannot exceed the number of failure groups with disks that can store data. The allowable values are 1, 2, and 3, with a default of 1.

If you want to change the data replication factor for the entire file system, the data disk in each storage pool must have a number of failure groups equal to or greater than the replication factor. For example, a failure with error messages occurs if you try to change the replication factor for a file system to 2 but the storage pool has only one failure group.

Maximum data replicas

The maximum number of copies of data blocks for a file can be specified at file system creation by using the **-R** option on the **mmcrfs** command. The default is 2. The allowable values are 1, 2, and 3, but cannot be less than the value of *DefaultDataReplicas*. This value cannot be changed.

File system management examples

In Example 4-79, we start with the `GPFS` file system that has only two NSDs in it.

Example 4-79 File system for test configuration

```
[root@fslinux1 gpfs]# mmdf GPFS
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 101 GB)
nsd02                10485760        1 Yes      Yes        10219008 ( 97%)           616 ( 0%)
nsd03                10485760        1 Yes      Yes        10219008 ( 97%)           616 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         20971520                              20438016 ( 97%)          1232 ( 0%)
```

```
             =============                      ==================== ====================
(total)             20971520                       20438016 ( 97%)          1232 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:           61754
Number of allocated inodes:      65792
Maximum number of inodes:        65792
```

To more fully secure the data, in this scenario we purchase new storage and activate the GPFS replica, for data and metadata. We configure two new disks on the OS and create two new NSDs on them. New NSDs have a different failure group number of initial disks. They are in a new storage system. The input file used to create the two new NSDs is named `nsd.file.dsk` and the contents are as shown in Example 4-80.

Example 4-80 The input file used to create two new NSDs using failure Group 10

```
%nsd:
  device=/dev/dm-2
  nsd=nsd04
  servers=fslinux1,fslinux2
  usage=dataAndMetadata
  failureGroup=10

%nsd:
  device=/dev/dm-5
  nsd=nsd05
  servers=fslinux1,fslinux2
  usage=dataAndMetadata
  failureGroup=10
```

We add two new NSDs to the file system by using the **mmadddisk** command as shown in Example 4-81.

Example 4-81 Using mmadddisk command to add disks

```
[root@fslinux1 gpfs]# mmadddisk GPFS -F nsd.file.dsk
The following disks of GPFS will be formatted on node fslinux2:
    nsd04: size 10240 MB
    nsd05: size 10240 MB
Extending Allocation Map
Checking Allocation Map for storage pool system
Completed adding disks to file system GPFS.
mmadddisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The **mmdf** command output is shown in Example 4-82.

Example 4-82 The mmdf command output to view new disks configured in the file system

```
[root@fslinux1 gpfs]# mmdf GPFS
[root@fslinux1 GPFS]# mmdf GPFS
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
```

```
-------------- ------------ -------- -------- ----- ------------------- ------------------
Disks in storage pool: system (Maximum disk size allowed is 101 GB)
nsd03                10485760        1 Yes      Yes        10065408 ( 96%)           840 ( 0%)
nsd02                10485760        1 Yes      Yes        10064896 ( 96%)           664 ( 0%)
nsd04                10485760       10 Yes      Yes        10265344 ( 99%)           648 ( 0%)
nsd05                10485760       10 Yes      Yes        10266112 ( 99%)           632 ( 0%)
               ------------                         ------------------- ------------------
(pool total)         41943040                              40661760 ( 97%)          2784 ( 0%)

               =============                        =================== ==================
(total)              41943040                              40661760 ( 97%)          2784 ( 0%)

Inode Information
-----------------
Number of used inodes:            4039
Number of free inodes:           61753
Number of allocated inodes:      65792
Maximum number of inodes:        65792
```

As you can see, new disks are added, they are free, and data remains on two old disks, nsd03 and nsd02. We configure the file system to have data and metadata replicated:

```
[root@fslinux1 GPFS]# mmchfs GPFS -m 2 -r 2
```

Now, configuration of the file system has been changed. The default option for data and metadata is 2, so we have two copies. However, data that was already in the file system is unchanged.

Because we have to force GPFS to properly replicate all files, according to the new configuration, we use the **-R** option with the **mmrestripefs** command, as shown in Example 4-83.

Example 4-83 Using mmrestripefs command to replicate all files

```
[root@fslinux1 GPFS]# mmrestripefs GPFS -R
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Fri Oct 24 16:57:22 2014  (     65792 inodes with total       392 MB
data processed)
Scan completed successfully.
```

Example 4-84 shows the output of the file system after the **mmrestripefs -R** command ends.

Example 4-84 The mmsf output after the mmrestripefs -R command runs

```
[root@fslinux1 GPFS]# mmdf GPFS
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
-------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 101 GB)
```

```
nsd03               10485760        1 Yes      Yes        10065664 ( 96%)           840 ( 0%)
nsd02               10485760        1 Yes      Yes        10064896 ( 96%)           648 ( 0%)
nsd04               10485760       10 Yes      Yes        10064640 ( 96%)           872 ( 0%)
nsd05               10485760       10 Yes      Yes        10065920 ( 96%)           616 ( 0%)
                -------------                             -------------------- -------------------
(pool total)         41943040                                  40261120 ( 96%)          2976 ( 0%)

                =============                             ==================== ===================
(total)              41943040                                  40261120 ( 96%)          2976 ( 0%)

Inode Information
-----------------
Number of used inodes:            4039
Number of free inodes:           61753
Number of allocated inodes:      65792
Maximum number of inodes:        65792
```

The data now is balanced across the file systems.

4.7.4 Exporting or importing a file system

GPFS allows you to move a GPFS file system definition from one cluster to another (an export-import function), or to delete a file system definition and save its configuration to be able to import the file system to a later time.

Two commands are available:

- The **mmexportfs** command saves the current file system definition in a file and exports it from the current cluster.
- The **mmimports** command imports the file system in a cluster where this file system is not already known.

The primary uses of these commands are to move a file system from one cluster to another or to mount a file system on another cluster, for example for backup purposes.

This section shows how these commands work.

The first cluster is named `mantest.usa` and consists of three nodes: slovenia, usa, and germany.

We create two new NSDs, nsddm2 and nsddm3, that are attached only to the node slovenia, which is a Linux node. On these two NSDs, we create a new file system named `testexportfs`. Example 4-85 shows the file system configuration.

Example 4-85 The mmdf, mmlsfs, and mmlsnsd commands to show file system configuration

```
g[root@fslinux1 .ssh]# mmdf GPFS
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 94 GB)
NSD001              10485760        1 Yes      Yes        10286080 ( 98%)           376 ( 0%)
NSD002              10485760        1 Yes      Yes        10286336 ( 98%)           376 ( 0%)
NSD003              10485760        1 Yes      Yes        10285312 ( 98%)           600 ( 0%)
                -------------                             -------------------- -------------------
(pool total)         31457280                                  30857728 ( 98%)          1352 ( 0%)
```

```
             =============                         ==================== ===================
(total)              31457280                          30857728 ( 98%)          1352 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:           61818
Number of allocated inodes:      65856
Maximum number of inodes:        65856
------------------------------------------------
[root@fslinux1 .ssh]# mmlsfs GPFS
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 8192                     Minimum fragment size in bytes
 -i                 4096                     Inode size in bytes
 -I                 16384                    Indirect block size in bytes
 -m                 1                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 1                        Default number of data replicas
 -R                 2                        Maximum number of data replicas
 -j                 cluster                  Block allocation type
 -D                 nfs4                     File locking semantics in effect
 -k                 all                      ACL semantics in effect
 -n                 32                       Estimated number of nodes that will mount file
system
 -B                 262144                   Block size
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota No                       Per-fileset quota enforcement
 --filesetdf        No                       Fileset df enabled?
 -V                 14.10 (4.1.0.4)          File system version
 --create-time      Wed Oct 22 16:20:08 2014 File system creation time
 -z                 No                       Is DMAPI enabled?
 -L                 4194304                  Logfile size
 -E                 Yes                      Exact mtime mount option
 -S                 No                       Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           Yes                      Fast external attributes enabled?
 --encryption       No                       Encryption enabled?
 --inode-limit      65856                    Maximum number of inodes
 --log-replicas     0                        Number of log replicas
 --is4KAligned      Yes                      is4KAligned?
 --rapid-repair     Yes                      rapidRepair enabled?
 --write-cache-threshold 0                   HAWC Threshold (max 65536)
 -P                 system                   Disk storage pools in file system
 -d                 NSD001;NSD002;NSD003     Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /shared                  Default mount point
 --mount-priority   0                        Mount priority
---------------------------------
[root@fslinux1 .ssh]# mmlsnsd
File system   Disk name    NSD servers
```

```
-------------------------------------------------------------------------
GPFS          NSD001       fslinux1,fslinux2
GPFS          NSD002       fslinux1,fslinux2
GPFS          NSD003       fslinux1,fslinux2
remotefs      NSD004       fslinux1,fslinux2
```

We are ready to export. The file system must be unmounted on all nodes to be exported because the definition is deleted from this cluster. If the file system uses any tiebreaker disk, you need to change the tiebreaker disk before continuing. We run the **mmexportfs** command, as shown in Example 4-86.

Example 4-86 Using mmexportfs to export file system definition and remove it from the cluster

```
[root@fslinux1 .ssh]# mmexportfs GPFS -o exportfs.out
mmexportfs: Processing file system GPFS ...
mmexportfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

This command creates a file named `exportfs.out` that contains the definition of the `GPFS` cluster file system. The definition is shown in Example 4-87.

Example 4-87 An example of mmexportfs output created file

```
[root@fslinux1 .ssh]# cat exportfs.out
%%9999%%:00_VERSION_LINE::1410:3:71::lc:fslinux2:fslinux1:0:/usr/bin/ssh:/usr/bin/scp:24971
46759570973977:lc2:1413408030::GPFS.Spectrum:2:1:1:2:A:::central:0.0:
%%home%%:10_NODESET_HDR:::2:TCP:AUTHONLY:1191:::13:1410:1410:L:4:::8:8:2:::::::
%%home%%:70_MMFSCFG::1:clusterName:GPFS.Spectrum:::::::::::::::::::::
%%home%%:70_MMFSCFG::2:clusterId:2497146759570973977:::::::::::::::::::
%%home%%:70_MMFSCFG::3:uidDomain:STORAGE:::::::::::::::::::::
%%home%%:70_MMFSCFG::4:dmapiFileHandleSize:32:::::::::::::::::::::
%%home%%:70_MMFSCFG::5:minReleaseLevel:1410 4.1.0.4:::::::::::::::::::
%%home%%:70_MMFSCFG::6:autoload:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::7:ccrEnabled:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::8:clusterManagerSelection:quorumNode:::::::::::::::::::
%%home%%:70_MMFSCFG::9:cipherList:AUTHONLY:::::::::::::::::::
%%home%%:70_MMFSCFG::10:tiebreakerDisks:NSD004:::::::::::::::::::
%%home%%:30_SG_HEADR:GPFS::150::::0::::no:::::::::::::::
%%home%%:40_SG_ETCFS:GPFS:1:%2Fshared:
%%home%%:40_SG_ETCFS:GPFS:2:    dev             = /dev/GPFS
%%home%%:40_SG_ETCFS:GPFS:3:    vfs             = mmfs
%%home%%:40_SG_ETCFS:GPFS:4:    nodename        = -
%%home%%:40_SG_ETCFS:GPFS:5:    mount           = mmfs
%%home%%:40_SG_ETCFS:GPFS:6:    type            = mmfs
%%home%%:40_SG_ETCFS:GPFS:7:    account         = false
%%home%%:50_SG_MOUNT:GPFS::rw:mtime:atime:::::::::::::::::::
%%home%%:60_SG_DISKS:GPFS:1:NSD001:0:1:dataAndMetadata:149DAC10543EF1ED:nsd:fslinux1,fslinu
x2::other::dmm:user::::::system:fslinux1,fslinux2:::::
%%home%%:60_SG_DISKS:GPFS:2:NSD002:0:1:dataAndMetadata:149DAC10543EF1F0:nsd:fslinux1,fslinu
x2::other::dmm:user::::::system:fslinux1,fslinux2:::::
%%home%%:60_SG_DISKS:GPFS:3:NSD003:0:1:dataAndMetadata:149DAC10543EF1F3:nsd:fslinux1,fslinu
x2::other::dmm:user::::::system:fslinux1,fslinux2:::::
```

Note: The `mmexportfs` output file must not be edited manually.

If you want this new file system on the same cluster, you have to import it, as shown in Example 4-88 on page 235.

Example 4-88 Using mmimports command to reimport the file system

```
[root@fslinux1 .ssh]# mmimportfs GPFS -i exportfs.out
mmimportfs: Processing file system GPFS ...
mmimportfs: Processing disk NSD001
mmimportfs: Processing disk NSD002
mmimportfs: Processing disk NSD003
mmimportfs: Committing the changes ...
mmimportfs: The following file systems were successfully imported:
        GPFS
mmimportfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Now, you are able to correctly mount the file system on this cluster, as before.

You might want to mount the file system to another cluster. Remember that the remote cluster *must* see all the disks that are part of this file system to be able to access them, and therefore to mount the file system.

In Example 4-89, we import a file system named `/dev/testexportfs` on the `testcluster` cluster, which is a single-node cluster. Because the local node is able to access the two disks where the file system is, importing is not a problem.

In this test, we import the file system on another cluster, as shown in Example 4-89.

Example 4-89 Importing file system on another cluster

```
# fdisk -l | grep dm
Disk /dev/dm-2: 26.8 GB, 26843545600 bytes
Disk /dev/dm-3: 26.8 GB, 26843545600 bytes

# mmimportfs /dev/testexportfs  -i testexportfsConfigEsported
mmimportfs: Processing file system testexportfs ...
mmimportfs: Processing disk nsddm2
mmimportfs: Incorrect node slovenia specified for command.
mmimportfs: Processing disk nsddm3
mmimportfs: Incorrect node slovenia specified for command.

mmimportfs: Committing the changes ...

mmimportfs: The following file systems were successfully imported:
        testexportfs
mmimportfs: The NSD servers for the following disks from file system testexportfs
were reset or not defined:
        nsddm2
        nsddm3
mmimportfs: Use the mmchnsd command to assign NSD servers as needed.
```

Important: Although Spectrum Scale correctly imports the file system, some modification to NSD might be necessary. The reason is because on source Spectrum Scale cluster, the two NSDs that belong to the `/dev/testexportfs` file system are configured to have a different node than the NSD server. However, this target cluster has only one node and the NSD servers node is not present. Therefore, Spectrum Scale resets the NSD server situation on these two NSDs. Now they appear similar to Example 4-90 on page 236.

Example 4-90 NSDs on the cluster as they appear after using the mmimportfs command

```
# mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
testexportfs  nsddm2       (directly attached)
testexportfs  nsddm3       (directly attached)
```

In our case, the file system import is acceptable because we have only one node in the cluster. However, if additional nodes are in the cluster, you must change the NSD server after the import process by using the **mmchnsd** command.

Now, we can mount the /dev/testexportfs file system, as shown in Example 4-91.

Example 4-91 Mount the imported file system

```
# mmmount /dev/testexportfs
Thu 24 18:22:11 EDT 2014: mmmount: Mounting file systems ...
# mmdf /dev/testexportfs
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 234 GB)
nsddm2               26214400       40 yes      yes        26128384 (100%)           488 ( 0%)
nsddm3               26214400       40 yes      yes        26128896 (100%)           456 ( 0%)
               -------------                         -------------------- -------------------
(pool total)         52428800                               52257280 (100%)           944 ( 0%)

               =============                         ==================== ===================
(total)              52428800                               52257280 (100%)           944 ( 0%)

Inode Information
-----------------
Number of used inodes:            4038
Number of free inodes:           48186
Number of allocated inodes:      52224
Maximum number of inodes:        52224
```

Another important point is that you can directly use the /var/mmfs/gen/mmsdrfs Spectrum Scale configuration file as input to the **mmimportfs** command to import a Spectrum Scale file system on a cluster. The reason is because the /var/mmfs/gen/mmsdrfs file contains all the cluster files, including the configuration of all file systems. Therefore, if you do not want the file system definition to also be removed from the source cluster, you can move the mmsdrfs file to a temporary directory of the destination cluster and use it with the **mmimportfs** command. Also, in this case, NSD servers are cleaned for every node that is unknown in the destination cluster.

Example 4-92 shows how you import a file system on another cluster by using only the mmsdrfs file.

Example 4-92 Importing a file system using the mmsdrfs file

```
fslinux1:/#scp /var/mmfs/gen/mmsdrfs fslinux2:/tmp/mmsdrfs.fslinux1
mmsdrfs
100%   11KB  11.2KB/s   00:00
```

After sending the `mmsdrfs` file, the file system can be imported on node *england* as shown in Example 4-93.

Example 4-93 Using the mmimportfs command on node england

```
# mmimportfs /dev/testexportfs  -i /tmp/mmsdrfs.fslinux1

mmimportfs: Processing file system testexportfs ...
mmimportfs: Processing disk nsddm2
mmimportfs: Incorrect node slovenia specified for command.
mmimportfs: Processing disk nsddm3
mmimportfs: Incorrect node slovenia specified for command.

mmimportfs: Committing the changes ...

mmimportfs: The following file systems were successfully imported:
        testexportfs
mmimportfs: The NSD servers for the following disks from file system testexportfs
were reset or not defined:
        nsddm2
        nsddm3
mmimportfs: Use the mmchnsd command to assign NSD servers as needed.
```

After the file system was imported, it can be mounted as shown in Example 4-94.

Example 4-94 Mounting the imported file system

```
# mmmount /dev/testexportfs
Thu 24 18:29:11 EDT 2014: mmmount: Mounting file systems ...

# df | grep /dev/testexportfs
/dev/testexportfs     52428800    171520  52257280   1% /testexportfs
```

Example 4-95 shows an extract of the `mmsdrf` file, and the `/dev/testexportfs` configuration.

Example 4-95 An extract of mmsdrfs showing the file system configuration

```
%home%%:4O_SG_ETCFS:testexportfs:1:%2Ftestexportfs:
%%home%%:4O_SG_ETCFS:testexportfs:2:    dev             = /dev/testexportfs
%%home%%:4O_SG_ETCFS:testexportfs:3:    vfs             = mmfs
%%home%%:4O_SG_ETCFS:testexportfs:4:    nodename        = -
%%home%%:4O_SG_ETCFS:testexportfs:5:    mount           = mmfs
%%home%%:4O_SG_ETCFS:testexportfs:6:    type            = mmfs
%%home%%:4O_SG_ETCFS:testexportfs:7:    account         = false
%%home%%:5O_SG_MOUNT:testexportfs::rw:mtime:atime::::::::::::::::::
%%home%%:6O_SG_DISKS:testexportfs:1:nsddm2:52428800:4O:dataAndMetadata:6585C0A84C3B8B44:nsd:slov
enia::other::dmm:user:::::::slovenia:::::
%%home%%:6O_SG_DISKS:testexportfs:2:nsddm3:52428800:4O:dataAndMetadata:6585C0A84C3B8B45:nsd:slov
enia::other::dmm:user:::::::slo
```

Both the **mmexportfs** and **mmimportfs** commands support the **all** option, which exports (or imports) all file systems at the same time. You use the **mmimportfs** command with the **-S** parameter to specify a file with the following format:

```
DiskName:ServerList
```

This format accepts one disk per line, followed by the new server list. In this way, NSD server configuration can be modified *during* the import operation so that you do not have to modify it *after* the import operating with the **mmchnsd** command.

Note: It is recommended to avoid direct use of the `mmsdrfs` files unless asked by your IBM representative.

4.8 Configuring IBM Spectrum Scale callback

Callback functionality can help you to automate certain tasks. Spectrum Scale is able to run user scripts (also referred to as *user exits)* when certain events occur. Using this functionality, it is possible to create your own executable scripts, which can be called automatically by Spectrum Scale after the defined event happens. For example, you can create a script to delete temporary old files when a low disk space condition is met. The following commands can be used to configure callback:

- **mmaddcallback**

  Sets up a new pair-event to monitor executable files.

- **mmlscallback**

  Lists already configured callbacks.

- **mmdelcallback**

  Deletes incorrect or old callback conditions.

Spectrum Scale can monitor two types of events:

- *Global events* trigger a callback on all nodes that are configured in the cluster.
- *Local events* trigger a callback script only on the node where the event occurs.

Global events are listed in Table 4-5.

Table 4-5 Global events

Global event	Description
nodeJoin	Triggered when one or more nodes join the cluster.
nodeLeave	Triggered when one or more nodes leave the cluster.
quorumReached	Triggered when a quorum has been established in the Spectrum Scale cluster.
quorumLoss	Triggered when a quorum has been lost in the Spectrum Scale cluster.
quorumNodeJoin	Triggered when one or more quorum nodes join the cluster.
quorumNodeLeave	Triggered when one or more quorum nodes leave the cluster.
clusterManagerTakeover	Triggered when a new cluster manager node has been elected. This event occurs when a cluster first starts or when the current cluster manager fails or resigns and a new node takes over as cluster manager.

Local events are listed in Table 4-6 on page 239.

Table 4-6 Local events

Local event	Description
lowDiskSpace	Triggered when the file system manager detects that disk space is running below the low threshold that is specified in the current policy rule.
noDiskSpace	Triggered when the file system manager detects that a disk ran out of space.
softQuotaExceeded	Triggered when the file system manager detects that a user or file set quota has been exceeded.
preMount, preUnmount, mount, unmount	Triggered when a file system is about to be mounted or unmounted or has been mounted or unmounted successfully. These events are generated for explicit **mount** or **unmount** commands, a remount after Spectrum Scale recovery, and a forced unmount when Spectrum Scale shuts down.
preStartup	Triggered after the Spectrum Scale daemon completes its internal initialization and joins the cluster, but before the node runs recovery for any VFS (Spectrum Scale-specific extensions) mount points that were already mounted, and before the node starts accepting user-initiated sessions.
startup	Triggered after a successful Spectrum Scale start and when the node is ready for user initiated sessions.
preShutdown	Triggered when Spectrum Scale detects a failure and is about to shut down.
shutdown	Triggered when Spectrum Scale completed the shutdown.
preRGRelinquish	Specifies that a Recovery Group is about to be relinquished.
postRGRelinquish	Specifies that a RecoveryGroup has been relinquished.
preRGTakeover	Specifies that a RecoveryGroup is about to be taken over.
postRGTakeover	Specifies that a RecoveryGroup has been taken over.
usageUnderSoftQuota	Triggered when quota usage falls below soft limits and grace time is reset.
diskFailure	Triggered when disk status changed to down but only on the Management Node.

Spectrum Scale also supports some predefined node classes and the node class value defines where the callback script should be triggered. When an event happens and there is *user exit* script registered for this event, Spectrum Scale invokes the script if the node where this event happened is on the node list that specified by the **-N** option from the **mmaddcallback** command. This option should be used to limit the number of nodes that should run the script.

The list of valid node classes values is displayed in Table 4-7.

Table 4-7 Valid node classes values

Node class value	Description
all	On all nodes.
quorumNodes	Quorum nodes only.

Node class value	Description
nonQuorumNodes	Non-quorum nodes only.
clientNodes	Non-manager nodes only.
managerNodes	Manager nodes only.
clusterManager	Cluster manager node only.
node name, node ip address	Single nodes specified by its name or ip_address.

After selecting a global or local event that you want Spectrum Scale to monitor, and writing your own script, add the callback by using the **mmaddcallback** command as shown in Example 4-96.

Example 4-96 mmaddcallback command syntax

```
mmaddcallback CallbackIdentifier --command CommandPathname --event
Event[,Event...] [--priority Value] [--async | --sync [--timeout Seconds]
[--onerror Action]] [-N {Node[,Node...] | NodeFile | NodeClass}] [--parms
ParameterString ...]
     or
  mmaddcallback {-S Filename | --spec-file Filename}
```

The command parameters are explained in *GPFS V4.1: Administration and Programming Reference,* SA23-1452-00.

Primarily, you set the following parameters:

- **CallbackIdentifier**, which is a name that you give to your callback
- **CommandPathname**, which is the complete path of your script
- **Event**, which indicates the event for which Spectrum Scale has to run your command

For this book, we configured and tested the callback functionality. Our setup and examples of how it works are described in Chapter 3, “Scenarios” on page 97 and Chapter 5, “Information lifecycle management” on page 245.

4.9 Monitoring IBM Spectrum Scale with SNMPD protocol

With Spectrum Scale, you can use the SNMPD protocol to monitor the status and the configuration of your cluster. It is possible to have one node, called the *collector* node, on which every SNMPD message is sent. To have this working properly, use the following steps:

1. The Net-SNMP master agent daemon (snmpd) must be installed on the collector node. The reason is because the Spectrum Scale subagent process connects to the snmpd.
2. On the master agent, the snmpd must be configured according to *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00:

   ```
   master agentx
   AgentXSocket tcp:localhost:705
   trap2sink managementhost
   ```

   In this configuration, *managementhost* is the host name or IP address of the host to which you want the SNMP traps to be sent.
3. If your Spectrum Scale cluster has many nodes or many file systems for which information must be collected, you must increase the timeout and retry parameters for communication between the Net-SNMP master agent and the Spectrum Scale subagent to allow time for

the volume of information to be transmitted. The `snmpd` configuration file entries for this are as follows:

```
agentXTimeout 60
agentXRetries 10
```

The entries have the following settings:

- `agentXTimeout` is set to 60 seconds for subagent to master agent communication
- `agentXRetries` is set to 10 for the number of communication retries

Other values might be appropriate depending on the number of nodes and file systems in your Spectrum Scale cluster.

4. After modifying the configuration file, restart the SNMP daemon.
5. Make the Spectrum Scale management information base (MIB) available on the nodes that have a management application installed (such as IBM Tivoli NetView®). The `/usr/lpp/mmfs/data/GPFS-MIB.txt` file contains the Spectrum Scale MIB.

   Perform the following steps:

   a. Copy the file into your SNMP MIB directory (usually `/usr/share/snmp/mibs`).

   b. Change your `snmpd.conf` file by adding the following line to enable the new MIB:

   ```
   mibs +GPFS-MIB
   ```

   c. Stop and restart snmpd to have it reload the configuration file.
6. After snmpd has been configured with Spectrum Scale MIB, the collector node must be declared in Spectrum Scale by using the **`mmchnode`** command, as follows:

   - To assign a collector node and start the SNMP agent:

     ```
     mmchnode --snmp-agent -N NodeName
     ```

   - To unassign a collector node and stop the SNMP agent:

     ```
     mmchnode --nosnmp-agent -N NodeName
     ```

   - To determine whether a Spectrum Scale SNMP subagent collector node is defined:

     ```
     mmlscluster |grep snmp
     ```

   - To change the collector node, issue the following two commands:

     ```
     mmchnode --nosnmp-agent -N OldNodeName
     mmchnode --snmp-agent -N NewNodeName
     ```

Some important considerations for the use of the SNMP-based monitoring capability:

- The SNMP collector node must be a Linux node in your Spectrum Scale cluster. Spectrum Scale uses Net-SNMP, which is not supported by AIX.
- The Net-SNMP master agent (also called the *SNMP daemon*, or *snmpd*) must be installed on the collector node to communicate with the Spectrum Scale subagent and with your SNMP management application. Net-SNMP is included in most Linux distributions and should be supported by your Linux vendor. Source and binary files for several platforms are available at this website:

  `http://net-snmp.sourceforge.net/download.html`
- If the monitored cluster is relatively large, you need to increase the communication timeout between the SNMP master agent and the Spectrum Scale SNMP subagent. In this context, a cluster is considered to be large if the number of nodes is greater than 25, or the number of file systems is greater than 15, or the total number of disks in all file systems is greater than 50.

For more information, see *Installing Net-SNMP* in the *GPFS V4.1: Advanced Administration Guide*, SC23-7032-00.

4.10 SSH configuration

Spectrum Scale recognizes the adminMode configuration parameter, which is set with the **mmchconfig** command and can have either of the following values:

- allToAll

  Indicates that every node in the cluster can be used to issue commands to all other nodes in the cluster. (This value is necessary for clusters that are running GPFS 3.2 or earlier.)

- central

  Indicates that a subset of nodes can be used by Spectrum Scale to issue remote commands to other nodes.

Regarding the SSH configuration, use allToAll to be sure that every node is authorized to all other nodes in the cluster to have everything properly working. If the adminMode is set to central, having all nodes SSH-authorized is not necessary, but we can have only a small set of designated nodes that can be trusted to run commands on their nodes without the password. Therefore, the SSH configuration is influenced by which method you choose to administer for your cluster.

When the SSH keys are generated for the first time on a node, for example with the **ssh-keygen -t** command for RSA or DSA, we create both private (id_rsa) and public (id_rsa.pub) keys for this node. To have an unattended SSH or SCP communication between nodes, the keys are created without a passphrase. To use SSH or SCP directly to another node, put the public key for the first node in the `authorized_keys` file, which is on the destination node. In this way, the first node is authorized to run remote commands to the secondary node. If you want all nodes to run remote commands on every other node, all nodes must have an `authorized_keys` file that contains all the public keys from all the nodes. Therefore, create an `authorized_keys` file with all the public keys and then use FTP, for example, to copy it to all the nodes.

To be sure that everything works correctly, try a remote command between nodes before the cluster is configured. If you want to use the *central* value of the **adminMode** parameter, you have to authorize all the nodes or only a subset of nodes by adding, into the `authorized_key` file, the SSH public keys of the nodes.

Another aspect to consider is that if a node is lost and must be reinstalled, you must re-create the SSH keys so that the node has new public keys. You must also remove the old SSH keys from the `authorized_key` file on the nodes and put newly re-created SSH key on the nodes.

Example 4-97 shows the SSH authorized_keys configuration on node tsmsrv. In this case, inside this file are public keys for all nodes of the cluster and nodes from remote clusters. Therefore, all nodes are authorized to enable remote commands on tsmsrv.

Example 4-97 Sample authorized_keys file

```
root@tsmsrv:/.ssh>cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCe4rrs3fj8Ezqee9pqNspehs2UOw9whLO9wXC2odjUmMoueEsUmTalycFyY9z
CLe6N6KPqX4h5/ngpgZBKGQ8MNtDSItgBXgRfE8OoWh5zdhYL1knZ4pRcNd+UPmOWYHhLRtRKlQoeSsuxRC7LQ4avNb
KNCWwEPzsGw6ci7nQEtk31V6B57yXStSKG81IOTtaj9Ed+izfYIAWxOyh15xOeRO8WqSZ32OqSpzDOFo+J9dqEYF9dS
jKw5rMWNi5J+nvS5u+UNnArLlnwVdWpGELJfEBKpS6FdFOvXr1Ktsvn1q7SPoyr6ekJaji493WhvFaReH+iC4V5GsEc
rOkLH8k7 root@fsaix1
```

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCSPuseb5e76FZGgjiaHmpkqI4PrxsCydbNuPt++JEzXusxkgmpEht6fU5dINc
MS/F3ic1/b79GtrtDg3ukYxY+MyOU4Vy+rFSjKDToOfJCY99mr/yJSj/BeNbdo9Gw1rCfs5PKVN/xxsXCXj8iv5N7CI
UuOjHN7BFnjlMpXYOw+k4ba2OP4XQ9m8N/u9S8dEdLmTXqfZi+yOjdOTjWsLIIvokeLGXI5CWo19I4K6yGCCM8RNYPG
Owu1JtYkjWD51kLkVpIbNCtgfmU2GtGQmt26j4qCjM1ctJrob52oqOLOWluBaR8cSFuKscjOQhUci43QnrAGo7V/ouT
tkRRlfqN root@fsaix2
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCXDf+ny5PAn5OdoeaNC6MIV+YlSRIosjufjBOTLhf6kH8y1vSnPPHCII+h5UW
OiB3drNjIZHTU+AUHXzfK4yCL44J2tPVc+RBzgJfHXnT7xhwALEdvvPHPuXu/zO/OXo/nVuxGNNXFawFO81OzIogryn
fAr8+sCnX5+s8nEtqZEy6vUXhBe6/uak1M9khiQqXL1OwLDa7dE7Lu9GRhw1rdXdwqoRp6QfifmHmmtGNC6w1MoJx1G
TE7Pi8AKyJwOzdUUhsZzkgvrIqzP5hgB89pEKFQaAJJfU1gGSVxPZniCJxFeSTJlmUMfOZ2UH4pEAhkIYZb3Go5nNI5
/wGXKLy3 root@fsaix4
ssh-dss
AAAAB3NzaC1kc3MAAACBANkFfrrpARarLeqmd/XdKFqGSdprWX/jUfx4t+Mhc/k4moV+9v8bwdDzlYEl2d8GICYdGu1
TRdWs5rL5L3sAa/hsongyyMkaOgBlwhPOBwpJFT9mV9JQqqf5wtB+P2bn8D/5vXZpRpPv2MrNAAk1LaSCQ+/F1cOOQf
mRkFKMh7DLAAAAFQDhkw39SgivvOY6hycRy9C6DPBs7wAAAIA35h8gGmMhPdqLVLObVONqNfRVeaOhrmeOrhuVaIk4w
LdIsXOsjbiPpSpPVMVyK28XOxOwmdYeEeBhl9EnVmbhMg1Wv/fSplWQ37vZrVT/seF7HgU7JiaO6FBc9FIpjlTaMmab
BH/zsnWdC4zc3ErQF1+qwhTBmjCU1Yz87/DUbQAAAIEAgtZ9lhQEneV2djJ9IR2Kzt+wfUkOGxavPuatGTUVZ/D1Ebh
JP8Y74XzFi9/fMc4p/fnXItciMO69bwizDZ4yhHqE4hKZ7qS7xNkCXNk/Sq6OYAJy9jtyE644SsBoTW+G9bxMUTBfia
ViWN386lehKodbNtCc8rDAWWMvxytNXLM= luibomar@oc1637416327.ibm.com
ssh-dss
AAAAB3NzaC1kc3MAAACBAPHwnNsU5M+OEPgqo5KNQQvLlnFEbQkKeUwSB9IXrQN+loJ/X3BYP5hVCIVAXyboP8ILbSZ
mbYKUozHihagMkHWvnHvVbgCJ64FnLxEW/VelkSEOUZOfYJNo6wpkJVJOMgTOGIp/dpOM5Xqzf3nv8z1uOenFkj5QZr
7PwFoJ5G9lAAAAFQCFB1M9K+fVzRhxtziwIZBqp56eYwAAAIA6oLRDUK89QO9HM5v1ZrtFoOxzUSynkVRBbttv/RB74
JJV6zAiFdnRcYl2LXeHdqyO36mK2/QF+TJkw6yEPK38FNTP8d/X+Sy+nJkbs6nTeM5TsVwx6RwVZUws57J2Lp9Q8nj5
AZhHecvr2UpQW3pxk4QR5ElzRoES5la5JhJb1wAAAIArGJ19lEFHHXkFhqYoVcEJTpnUBL+EIek5dtCu9UQwB96jZzE
f9OM3ulFOQsmVW7kxKdOopLOrAGIz9pylPOF5nqwkgtLaXEMOKtTVdIMvsFtDfzyJg5F1vZURt7+ISWdVfrmcGfjlvg
Gc9rxC1Ow/Gu+YY/6QouchXGIiKOTw6Q== root@fslinux1
ssh-dss
AAAAB3NzaC1kc3MAAACBAIKK5G5ckX1LYQzqNKAxOfDG1S2wMn4jGqy3Sw4CrRYq2OdXU3ysorYC7ab1iDIKq4gk58N
2d8hnwweKhbK8LqfzDbicoteAvl8puC7KTLA4S2KSc9NyH1bXr/2acKeWqzzy9HsCcOqifhaa8/bGNO6Jn/tC7oJI5d
Eie6GAqT/nAAAAFQCmSaNHj2UDOgZ7YB9HkgK2/OS/lwAAAIAEtLQ34IjzBRSuUFmBx8gu8iFaoeaTyfGxBP4NBFb5b
O9d1P6u4IqyCYse2LfqoONUheM4DdtOB3vBnqLECp+G31+xzlEIqZUDxqkhGkopdZGUHP2rSk2XkCL718+q93Izh/gP
NP42r5nXC8gbVmmBHY3MepBZHvEiUb9cCI5WugAAAIBrqjsCxUFUoGUVgzvz1roioe5eOBTpPVsuUmdR/gUj4SurVhr
N6HUYCfNf4dGhVkoSxZKFA/9P1V9zUGYKJ1SompZXPNVqUhvkzJUZgK+jUv3851W5iBSmgIcF6arHilpeAWigHUvwVz
cx1E7vdOMo611LMzJhY6Kh/lkHYuHwVw== root@fslinux2
```

5

Information lifecycle management

This chapter explains the technology and features that are used by IBM Spectrum Scale to manage the information over its lifetime.

The chapter also shows the tools that are provided by IBM Spectrum Scale for managing the information. And the benefits and possibilities of each of those tools.

The following sections are described in this chapter:

- Explaining the ILM concept
- Fileset
- Snapshot
- Storage pools
- Immutability and appendOnly attributes
- Quotas
- Policies and rules
- Access control list
- The mmfind policy sample
- Differences with IBM EasyTier

Note: Information lifecycle management (ILM) is only available with IBM Spectrum Scale Standard Edition or higher.

© Copyright IBM Corp. 2015. All rights reserved.

5.1 Explaining the ILM concept

IBM Spectrum Scale can help you achieve ILM efficiencies through powerful policy driven automated tiered storage management. The Spectrum Scale ILM toolkit helps you manage sets of files and pools of storage, and also enables you to automate the management of file data.

Spectrum Scale introduces several terms that are used regarding ILM. Several terms are also used in other software or hardware products, and can have similar or different meaning. To reduce the confusion, this section explains the terms and shows several examples.

5.2 Fileset

In most file systems, a file hierarchy is represented as a series of directories that form a tree-like structure. Each directory contains other directories, files, or other file system objects such as symbolic links and hard links. Every file system object has a name associated with it, and is represented in the namespace as a node of the tree.

In addition, Spectrum Scale uses a file system object that is called a *fileset*. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. Filesets provide a means of partitioning the file system to allow administrative operations at a finer granularity than the entire file system:

- Filesets can be used to define quotas on both data blocks and inodes.
- The owning fileset is an attribute of each file and can be specified in a policy to control initial data placement, migration, and replication of the file's data.
- Fileset snapshots can be created instead of creating a snapshot of an entire file system.

Spectrum Scale supports independent and dependent filesets. An *independent* fileset is a fileset with its own inode space. An *inode space* is a collection of inode number ranges reserved for an independent fileset. An inode space enables more efficient per-fileset functions, such as fileset snapshots. A *dependent fileset* shares the inode space of an existing, independent fileset. Files that are created in a dependent fileset are assigned inodes in the same collection of inode number ranges that were reserved for the independent fileset from which it was created.

When the file system is created, only one fileset, called the *root fileset*, exists. It contains the root directory as well as any system files such as quota files. As new files and directories are created, they automatically become part of the parent directory's fileset. The fileset to which a file belongs is largely transparent for ordinary file access, but the containing fileset can be displayed along with the other attributes of each file using the **`mmlsattr -L`** command.

The root directory of a Spectrum Scale file system is also the root of the root fileset.

Figure 5-1 on page 247 shows an example of several file sets, as follows:

- The `Fileset_u1` starts from the `/dir4/dir_b` directory and has `/user1/` as the parent directory.
- The `Fileset_u2`, which starts at `/user2` directory and has as its children the `data1`, `data2`, `data3`, and `data4` directories.

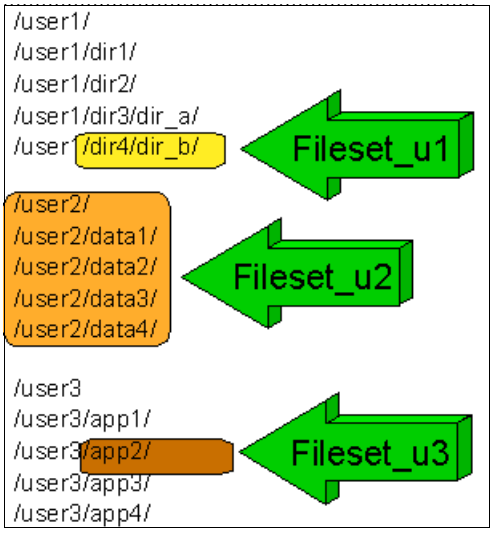


Figure 5-1 Fileset example

Ordinary file access is largely transparent regardless of which fileset the file belongs to. The **mmlsattr -L** command shows to which fileset a particular file belongs as shown in Example 5-1.

Example 5-1 Showing the belonging fileset for a file

```
#mmlsattr -L myfile1
file name:            myfile1
metadata replication: 1 max 3
data replication:     1 max 3
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         myFileset
snapshot name:
creation time:        Sat Oct 10 09:59:58 2014
Windows attributes:   ARCHIVE
Encrypted:            no
```

Note: The Spectrum Scale root file system is an independent fileset that cannot be deleted or unlinked.

Instead of creating a global snapshot of an entire file system, a fileset snapshot can be created to preserve the contents of a single independent fileset plus all dependent filesets that share inode space.

Example 5-2 shows how to create an independent fileset, and Example 5-3 on page 248 shows how to create a dependant fileset by using the **mmcrfileset** command. The **-t** option can be used to add comments to the fileset that is being created.

Example 5-2 Creating an independent fileset

```
#mmcrfileset GPFS independentFSET --inode-space new -t MyIndependantFileset
Fileset independentFSET created with id 1 root inode 262147.
#
```

Example 5-3 Creating a dependent fileset

```
#mmcrfileset device dependantFSET --inode-space independentFSET -t MyDependantFileset
Fileset dependantFSET created with id 4 root inode 266496.
#
```

When created, filesets need to be linked into the file system structure. A newly created fileset is not visible to the user until it is attached to the namespace by issuing the **mmlinkfileset** command. Filesets are attached to the namespace with a special link called a *junction*. The junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset, the parent, to the root directory of a child fileset. From the user's viewpoint, a junction always appears as though it were a directory, but the user is not allowed to issue the **unlink** or **rmdir** commands on a junction. Instead, the **mmunlinkfileset** command must be used to remove a junction.

When linking a fileset, consider the following factors:

- The file system must be mounted in order to link a fileset.
- An independent fileset can be linked into only one location anywhere in the namespace.
- A dependent fileset can only be linked inside its own inode space.
- If **JunctionPath** parameter is not specified, the junction is created in the current directory and has the same name as the fileset being linked.
- The user can use the **mv** command on the directory to move to a new location in the parent fileset, but the **mv** command is not allowed to move the junction to a different fileset.

To link both independent filesets, the **mmlinkfileset** command is used as shown in Example 5-4.

Example 5-4 Linking an independent fileset

```
#mmlinkfileset GPFS independentFSET -J INDEP_FSET
Fileset independentFSET linked at /GPFS/INDEP_FSET
#
```

If no -J option is passed, the fileset is linked on the current directory where the command is run on a directory with the same name as the fileset name that is being linked. See Example 5-5.

Example 5-5 Linking an independent file set if no -J option is passed

```
# mmlinkfileset GPFS independentFSET
Fileset independentFSET linked at /GPFS/independentFSET
#
```

After the independent fileset has been linked, the user can start using it normally.

To link an independent fileset, the junction must be in the inode space where the fileset belongs to. Example 5-6 shows mounting the independent fileset that was created on Example 5-2 on page 247.

Example 5-6 Linking a fileset to a junction

```
# mmlinkfileset GPFS dependantFSET -J /GPFS/INDEP_FSET/DEPENDANT_FSET
Fileset dependantFSET linked at /GPFS/INDEP_FSET/DEPENDANT_FSET
```

If you try to mount a dependant fileset out of the inode space where it belongs, you would see the error that is shown in Example 5-7.

Example 5-7 Error when trying to mount a dependent fileset out of the inode space where it belongs

```
#mmlinkfileset GPFS dependantFSET -J /GPFS/myFileset/dependantFSET
Fileset dependantFSET can only be linked within its own inode space.
mmlinkfileset: Command failed. Examine previous error messages to determine cause.
```

Once the dependent fileset has been linked, the user can start using it normally. See the status of the fileset with the **mmlsfileset** command as shown in Example 5-8.

Example 5-8 Listing the fileset

```
# mmlsfileset GPFS
Filesets in file system 'GPFS':
Name                    Status    Path
root                    Linked    /GPFS
myFileset               Linked    /GPFS/myFileset
independentFSET         Linked    /GPFS/INDEP_FSET
dependantFSET           Linked    /GPFS/INDEP_FSET/DEPENDANT_FSET
```

To unlink a fileset from the file system is to remove the junction of the fileset. The junction can be specified by path or by naming the fileset that is its target. The unlink fails if there are files open in the fileset, unless the **-f** flag is specified. To unlink a fileset, the **mmunlinkfileset** command is used as shown in Example 5-9 and in Example 5-10.

Example 5-9 Unlinking a fileset by fileset name

```
#mmunlinkfileset GPFS dependantFSET
Fileset dependantFSET unlinked.
#mmlsfileset GPFS
Filesets in file system 'GPFS':
Name                    Status    Path
root                    Linked    /GPFS
myFileset               Linked    /GPFS/myFileset
independentFSET         Linked    /GPFS/independentFSET
dependantFSET           Unlinked  --
#
```

See Example 5-10 to unlink a fileset by Junction name.

Example 5-10 Unlinking a fileset by Junction name

```
#mmunlinkfileset GPFS -J /GPFS/independentFSET
Filesets in file system 'GPFS':
Name                    Status    Path
root                    Linked    /GPFS
myFileset               Linked    /GPFS/myFileset
independentFSET         Unlinked  --
dependantFSET           Unlinked  --
#
```

For both of the preceding examples, the unlinking can be forced by appending the `-f` option at the end of the command.

Note: It is possible to unlink an independent fileset while dependent filesets are still linked inside of its space. By doing so, all filesets in the same inode space are not accessible until the parent fileset is linked. When the parent fileset is linked, the children ones that are not unlinked are again accessible.

To change a fileset's junction, you have to first unlink the fileset using the **mmunlinkfileset** command, and then create the new junction using the **mmlinkfileset** command. To change the attributes of an existing fileset, including the fileset name, use the **mmchfileset** command. In Example 5-11, the **mmchfileset** command is used to increase the number of maximum inodes for the fileset.

Example 5-11 Changing the maximum number of inodes in a fileset

```
#mmlsfileset GPFS independentFSET -i
Collecting fileset usage information ...
Filesets in file system 'GPFS':
Name                    Status    Path      InodeSpace MaxInodes AllocInodes UsedInodes
independentFSET         Unlinked  --        2          100032    65856       1

# mmchfileset GPFS independentFSET --inode-limit 100999
Set maxInodes for inode space 2 to 101056
Fileset independentFSET changed.
#

#mmlsfileset GPFS independentFSET -i
Collecting fileset usage information ...
Filesets in file system 'GPFS':
Name                    Status    Path      InodeSpace MaxInodes AllocInodes UsedInodes
independentFSET         Unlinked  --        2          101056    65856       1
#
```

To delete a fileset, use the **mmdelfileset** command. When deleting a fileset, consider the following points:

- Root fileset cannot be deleted.
- A fileset that is not empty cannot be deleted unless the **-f** flag is specified.
- A fileset that is linked into the name space cannot be deleted until it is unlinked with the **mmunlinkfileset** command.
- A dependent fileset can be deleted at any time.
- An independent fileset cannot be deleted if it has any dependent filesets or fileset snapshots.
- Deleting a dependent fileset that is included in a fileset or global snapshot removes it from the active file system, but it remains part of the file system in a deleted state.
- Deleting an independent fileset that is included in any global snapshots removes it from the active file system, but it remains part of the file system in a deleted state.
- A fileset in the deleted state is displayed in the **mmlsfileset** command output with the fileset name in parenthesis. If the **-L** flag is specified, the latest including snapshot is also displayed. The **--deleted** option of the **mmlsfileset** command can be used to display only deleted filesets.
- The contents of a deleted fileset are still available in the snapshot, through some path name containing a *.snapshots* component because it was saved when the snapshot was created.

- When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

In Example 5-12, the **mmdelfileset** command is used to delete a snapshot, and the **mmlsfileset** command is used to display additional information for the fileset.

Example 5-12 Deleting a fileset and listing the deleted fileset belonging to a global snapshot

```
# mmdelfileset GPFS independentFSET -f
Checking fileset ...
Fileset contains a dependent fileset 'dependantFSET'.
Fileset independentFSET cannot be deleted.
mmdelfileset: Command failed. Examine previous error messages to determine cause.
# mmdelfileset GPFS dependantFSET -f
Checking fileset ...
Checking fileset complete.
Deleting fileset ...
Fileset dependantFSET deleted.
# mmdelfileset GPFS independentFSET
Checking fileset ...
Checking fileset complete.
Deleting fileset ...
Fileset independentFSET deleted.
#mmlsfileset GPFS -L --deleted
Filesets in file system 'GPFS':
Name             Id RootInode  ParentId   Created                  InodeSpace MaxInodes AllocInodes Comment
(independentFSET) 3  latest:    globasnp01 Sat Oct 11 01:02:42 2014 2          101056    65856
```

Note: IBM Tivoli Storage Manager tracks files by path name and does not track file sets. As a result, when you unlink a file set, it appears to Tivoli Storage Manager that you deleted the contents of the file set. Therefore, the IBM Tivoli Storage Manager Backup Archive client deactivates the data on the Tivoli Storage Manager server, which might result in the loss of backup data during the expiration process.

When dealing with backups and restores for filesets, look at Chapter 7, “Backup and disaster recovery using IBM Spectrum Scale” on page 345 to also back up the fileset structures and metadata, and how to do bare metal restores.

5.3 Snapshot

IBM Spectrum Scale provides the functionality to create snapshots at the file system, file set, and file level. Each Spectrum Scale file system can have multiple snapshots of any of the types at the same time.

5.3.1 Snapshot at the file system level

A snapshot of an entire Spectrum Scale file system can be created to preserve the contents of the file system at a single point in time. Snapshots of the entire file system are also known as *global snapshots*. The storage overhead for maintaining a snapshot is keeping a copy of data blocks that would otherwise be changed or deleted after the time of the snapshot.

Snapshots of a file system are read-only; changes can only be made to the active (that is, normal, non-snapshot) files and directories.

The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time that the snapshot was created. Snapshots also provide an online backup capability that allows easy recovery from common problems such as accidental deletion of a file, and comparison with older versions of a file.

Note: Snapshots are not copies of the entire file system. Do not use snapshots as protection against media failures or backup solution.

Use the **mmcrsnapshot** command to create a snapshot of an entire Spectrum Scale file system at a single point in time. Global snapshots also include all the filesets in the file system. To create a global snapshot, use the **mmcrsnapshot** command as shown in Example 5-13.

Example 5-13 Creating a global snapshot

```
# mmcrsnapshot GPFS globasnp01
Flushing dirty data for snapshot globasnp01...
Quiescing all file system operations.
Snapshot globasnp01 created with id 2
```

Global snapshots appear in a subdirectory in the root directory of the file system, whose default name is `.snapshots`. If you prefer to access snapshots from each directory rather than traversing through the root directory, you can use an invisible directory to make the connection by issuing the **mmsnapdir** command as shown in Example 5-14.

Example 5-14 Linking global snapshot to all directories

```
# pwd
/GPFS/myFileset/.snapshots
# ls -al
total 1
dr-xr-xr-x 4 root root 8192 Oct 11 02:04 .
drwx------ 3 root root 4096 Oct 11 02:02 ..
drwx------ 2 root root 4096 Oct 11 00:23 snap01
# mmsnapdir GPFS -r --show-global-snapshots allfilesets
# ls -al
total 1
dr-xr-xr-x 4 root root 8192 Oct 11 02:04 .
drwx------ 3 root root 4096 Oct 11 02:02 ..
drwx------ 2 root root 4096 Oct 11 01:05 globasnp01
drwx------ 3 root root 4096 Oct 11 02:02 globasnp02
drwx------ 2 root root 4096 Oct 11 00:23 snap01
```

Note: The directories that are added by the **mmsnapdir** command are invisible to the **ls** command or to the `readdir()` function. This technique prevents recursive file system utilities, such as **find** or **tar**, from entering into the snapshot tree for each directory they process.

If you prefer to access global snapshots from the root directory, use the **mmsnapdir** command with the **--show-global-snapshots rootfileset** option as shown in Example 5-15.

Example 5-15 Unlinking global snapshot from all directories

```
# pwd
/GPFS/myFileset/.snapshots
# ls -al
total 1
dr-xr-xr-x 4 root root 8192 Oct 11 02:04 .
```

```
drwx------ 3 root root 4096 Oct 11 02:02 ..
drwx------ 2 root root 4096 Oct 11 01:05 globasnp01
drwx------ 3 root root 4096 Oct 11 02:02 globasnp02
drwx------ 2 root root 4096 Oct 11 00:23 snap01
# mmsnapdir GPFS -r --show-global-snapshots rootfileset
# ls -al
total 1
dr-xr-xr-x 4 root root 8192 Oct 11 02:04 .
drwx------ 3 root root 4096 Oct 11 02:02 ..
drwx------ 2 root root 4096 Oct 11 00:23 snap01
```

It is also possible to change the default `.snapshot` directory with the **mmsnapdir** command as shown in Example 5-16.

Example 5-16 Changing the snapshot directory for global and filesets

```
# mmsnapdir GPFS --global-snapdir cave
# mmsnapdir GPFS --global-snapdir cave
# ls -al /GPFS/cave/
total 769
dr-xr-xr-x 4 root root   8192 Oct 11 02:04 .
drwxr-xr-x 4 root root 262144 Oct 11 02:03 ..
drwxr-xr-x 3 root root 262144 Oct 11 01:26 globasnp01
drwxr-xr-x 4 root root 262144 Oct 11 02:03 globasnp02
# ls -al /GPFS/myFileset/snapdir/
total 1
dr-xr-xr-x 4 root root 8192 Oct 11 02:04 .
drwx------ 3 root root 4096 Oct 11 02:02 ..
drwx------ 2 root root 4096 Oct 11 00:23 snap01#
```

To see the current snapshot settings, run the **mmsnapdir** command as shown in Example 5-17.

Example 5-17 Listing the snapshot settings

```
# mmsnapdir GPFS -q
Fileset snapshot directory for "GPFS" is "snapdir" (root directory only)
Global snapshot directory for "GPFS" is "cave" in root fileset
```

To list the global snapshots of a certain file system, the status, sizes, and creation date, the **mmlssnapshot** command is used as shown in Example 5-18.

Example 5-18 Listing all global snapshots in the file system

```
# mmlssnapshot GPFS -s global
Snapshots in file system GPFS:
Directory               SnapId    Status  Created                   Fileset
globasnp01              2         Valid   Sat Oct 11 01:47:44 2014
globasnp02              3         Valid   Sat Oct 11 02:04:05 2014
```

To include the size in kilobytes used by each snapshot, the **-d** parameter needs to be used as shown in Example 5-19.

Example 5-19 The -d parameter is used to include the size in kilobytes used by each snapshot

```
# mmlssnapshot GPFS -d -s global
Snapshots in file system GPFS: [data and metadata in KB]
Directory               SnapId    Status  Created                   Fileset          Data  Metadata
globasnp01              2         Valid   Sat Oct 11 01:47:44 2014                    256      1280
globasnp02              3         Valid   Sat Oct 11 02:04:05 2014                      0       256
```

It is possible to select the size output in megabytes (MB), gigabytes (GB), or terabytes (TB) with the **--block-size** parameter. In Example 5-20, MB is selected.

Example 5-20 Selecting the size output in MB with the --block-size parameter

```
# mmlssnapshot GPFS -d --block-size M -s global
Snapshots in file system GPFS: [data and metadata in MB]
Directory               SnapId    Status  Created                   Fileset           Data  Metadata
globasnp01              2         Valid   Sat Oct 11 01:47:44 2014                      1         2
globasnp02              3         Valid   Sat Oct 11 02:04:05 2014                      0         1
```

When a global snapshot is no longer needed or we want to free the space used by the global snapshot, delete the global snapshot. To delete a global snapshot, use the **mmdelsnapshot** command as shown in Example 5-21.

Example 5-21 Deleting a global snapshot

```
# mmdelsnapshot GPFS globasnp02
Invalidating snapshot files in globasnp02...
Deleting files in snapshot globasnp02...
 100.00 % complete on Sat Oct 11 02:20:41 2014  (    197504 inodes with total          0 MB data processed)
Invalidating snapshot files in globasnp02/F/...
Delete snapshot globasnp02 complete, err = 0
```

Note: It is possible to specify which nodes participate in the deletion of the snapshot.

If you need to restore to the point of the global snapshot, and you need only some me files or directories but not the whole file system, you can do so by navigating through the snapshot and copying the files and directories that you need to the active file system.

If you need to restore to the point of a global snapshot the whole file system, run the **mmrestorefs** command. Example 5-22 on page 255 shows how to restore a file system from a global snapshot.

When restoring from a global file system, the following considerations need to be assessed:

- The file system must be unmounted in all nodes in the cluster before running the **mmrestorefs** command and until the **mmrestorefs** command has completed successfully. The file system can be mounted if the **mmrestorefs** command has not completed successfully but was run with the **-c** parameter. The **-c** option indicates that the **mmrestorefs** command must continue to restore the file system if errors occur.
- Automatic quota activation upon mounting the file system is not restored by the **mmrestorefs** command. You must issue the **mmchfs -Q** *yes* command to restore automatic quota activation.
- Snapshots are not affected by the **mmrestorefs** command. Consequently, a failure while restoring one snapshot may possibly be recovered by restoring a different snapshot.
- When the **mmsnapdir -a** option (add a snapshots subdirectory to all subdirectories in the file system) is in effect, the snapshots subdirectories may no longer show the complete list of snapshots containing the parent directory. This occurs if the file system was restored from a snapshot that was not the latest. Since the root directory is contained in all snapshots, its `snapshots` subdirectory will always show the complete list of snapshots.
- Consider if the file system is encrypted because the files that were removed after the snapshot was taken are restored with the same encryption attributes of the file in the snapshot. Unless **--preserve-encryption-attributes** option is issued with the command, the file is re-created with the encryption policy in place at the time the file is restored.

Example 5-22 Restoring a file system from a global snapshot

```
# mmrestorefs GPFS globasnp01
[2014-10-11 02:22:01] Prepare restore ...
[2014-10-11 02:22:08] Scan inodes ...
[2014-10-11 02:22:16] 328000 inodes have been scanned. 50% of total.
[2014-10-11 02:22:24] 656000 inodes have been scanned. 100% of total.
[2014-10-11 02:22:24] Construct operation list ...
[2014-10-11 02:22:24] 0 operations have been added to list.
[2014-10-11 02:22:24] 3 operations have been added to list.
[2014-10-11 02:22:24] Restore files ...
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     892672        5242880    17.026367188%
[I] 4034 of 266880 inodes used: 1.511541%.
[I] Loaded policy rules from /GPFS/RestoreTemp16345/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:22:24 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -o globasnp01
-t /GPFS/RestoreTemp16345 -B 100  -v 1   -p 3 -c 16345'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/RestoreTemp16345/opList...
[I] 2014-10-10@23:22:24.985 Executing file list: /GPFS/RestoreTemp16345/opList. 3 files dispatched.
[I] A total of 3 PDRs from filelist /GPFS/RestoreTemp16345/opList have been processed; 0 'skipped' records
and/or errors.
[I] A total of 3 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
   0 'skipped' files and/or errors.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     892672        5242880    17.026367188%
[I] 4033 of 266880 inodes used: 1.511166%.
[I] Loaded policy rules from /GPFS/RestoreTemp16345/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:22:25 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -o globasnp01
-t /GPFS/RestoreTemp16345 -B 100  -v 1   -p 51 -c 16345'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/RestoreTemp16345/opList.copy...
[I] 2014-10-10@23:22:25.530 Executing file list: /GPFS/RestoreTemp16345/opList.copy. 0 files dispatched.
[I] A total of 0 PDRs from filelist /GPFS/RestoreTemp16345/opList.copy have been processed; 0 'skipped'
records and/or errors.
[I] A total of 0 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
   0 'skipped' files and/or errors.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     892672        5242880    17.026367188%
[I] 4033 of 266880 inodes used: 1.511166%.
[I] Loaded policy rules from /GPFS/RestoreTemp16345/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:22:25 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -o globasnp01
-t /GPFS/RestoreTemp16345 -B 100  -v 1   -p 52 -c 16345'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
```

```
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/RestoreTemp16345/opList.copy.delta...
[I] 2014-10-10@23:22:26.064 Executing file list: /GPFS/RestoreTemp16345/opList.copy.delta. 0 files
dispatched.
[I] A total of 0 PDRs from filelist /GPFS/RestoreTemp16345/opList.copy.delta have been processed; 0
'skipped' records and/or errors.
[I] A total of 0 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
   0 'skipped' files and/or errors.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     892672        5242880    17.026367188%
[I] 4033 of 266880 inodes used: 1.511166%.
[I] Loaded policy rules from /GPFS/RestoreTemp16345/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:22:26 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -o globasnp01
-t /GPFS/RestoreTemp16345 -B 100  -v 1   -p 6 -D -c 16345'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/RestoreTemp16345/dir.new...
[I] 2014-10-10@23:22:26.624 Executing file list: /GPFS/RestoreTemp16345/dir.new. 1 files dispatched.
[I] A total of 1 PDRs from filelist /GPFS/RestoreTemp16345/dir.new have been processed; 0 'skipped' records
and/or errors.
[I] A total of 1 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
   0 'skipped' files and/or errors.
[2014-10-11 02:22:26] Restore completed.
[2014-10-11 02:22:26] Clean up.#
```

Note: The **mmrestorefs** command can take a long time to process, depending on the file system size. It is possible to define the number of threads, files per thread, and nodes for this operation from the defaults of 24 threads, 60 files per thread, and all nodes.

5.3.2 Snapshot at the fileset level

To create a snapshot of a fileset, use the **mmcrsnapshot** command as shown in Example 5-23.

Example 5-23 Creating a fileset snapshot

```
# mmcrsnapshot GPFS myfsetsnap01 -j myFileset
Flushing dirty data for snapshot myFileset::myfsetsnap01...
Quiescing all file system operations.
Snapshot myFileset::myfsetsnap01 created with id 4.#
```

Note: If an independent fileset has dependent filesets that share its inode space, a snapshot of the independent fileset will also include those dependent filesets. In other words, a fileset snapshot is a snapshot of the whole inode space.

Fileset snapshots appear in a similar to root global snapshot `.snapshots` subdirectory in the root directory of each independent fileset.

If you prefer to access the snapshots from each directory rather than traversing through the root directory, you can use an invisible directory to make the connection by issuing the **mmsnapdir** command with the **-a** option as shown on the global snapshot in Example 5-14 on page 252.

If you prefer to access global snapshots from the root directory of all independent filesets, use the **mmsnapdir** command with the **--show-global-snapshots allfilesets** option as shown in Example 5-14 on page 252. With this option, global snapshots also appear in the `snapdir` in the fileset root directory.

The global snapshots also appear in the `snapdirs` in each non-root directory if “all-directories” (the **-a** option shown in Example 5-14 on page 252) is enabled.

To return to the default setting, use **--show-global-snapshots rootfileset** as shown on Example 5-15 on page 252, and global snapshots will only be available in root of the file system, or the root fileset, if “all-directories” is enabled.

To list the global snapshots of a certain fileset, the status, sizes, and creation date, the **mmlssnapshot** command is used as shown in Example 5-24.

Example 5-24 Listing snapshots of a fileset with size data

```
# mmlsfileset GPFS  myFileset -d
Collecting fileset usage information ...
Filesets in file system 'GPFS':
Name                    Status    Path                                    Data (in KB)
myFileset               Linked    /GPFS/myFileset                                512#
```

To delete a fileset snapshot, use the **mmdelsnapshot** command as shown in Example 5-25.

Example 5-25 Deleting a fileset snapshot

```
# mmdelsnapshot GPFS myfsetsnap01
File system GPFS does not contain global snapshot myfsetsnap01, err = 2
Delete snapshot myfsetsnap01 complete, err = 2
mmdelsnapshot: Command failed. Examine previous error messages to determine cause.
# mmdelsnapshot GPFS myfsetsnap01 -j myFileset
Invalidating snapshot files in myFileset::myfsetsnap01...
Deleting files in snapshot myFileset::myfsetsnap01...
 100.00 % complete on Sat Oct 11 02:36:17 2014  (     65856 inodes with total          0 MB data processed)
Invalidating snapshot files in myFileset::myfsetsnap01/F/...
Delete snapshot myFileset::myfsetsnap01 complete, err = 0#
```

Note: It is possible to specify which nodes participate in the deletion of the snapshot.

Independent fileset snapshot data and attribute files can be restored by using the **mmrestorefs** command. When a restore from a global file system is going to be performed, the following considerations need to be assessed:

- The file system must be mounted and the fileset linked from nodes in the cluster that are to participate in the restore. The **-N**, **--threads**, **--work-unit, --suppress-external-attributes**, **--preserve-encryption-attributes**, and **--log-quiet** options of the **mmrestorefs** command are specific to restoring independent fileset snapshots. It is preferable to run the **mmrestorefs** command when the system is idle and no other commands are running. While the restore is in progress, do not unlink the fileset, unmount the file system, or delete the fileset, fileset snapshot, or file system.
- Snapshots are not affected by the **mmrestorefs** command. Consequently, a failure while restoring one snapshot may possibly be recovered by restoring a different snapshot.
- When the **mmsnapdir -a** command option (add a snapshots subdirectory to all subdirectories in the file system) is in effect, the snapshots subdirectories can no longer show the complete list of snapshots containing the parent directory. This occurs if the file system was restored from a snapshot that was not the latest. Since the root directory is

contained in all snapshots, its snapshots subdirectory will always show the complete list of snapshots.

- Consider if the file system is encrypted because the files that were removed after the snapshot was taken are restored with the same encryption attributes of the file in the snapshot. Unless the **--preserve-encryption-attributes** option is issued with the command, the file is re-created with the encryption policy in place at the time the file is restored.

Note: Snapshots are not copies of the entire fileset. Do not use snapshots as protection against media failures or backup solution.

Example 5-26 shows the **mmrestorefs** command.

Example 5-26 Restoring a fileset from a snapshot

```
# mmrestorefs GPFS myfsetsnap01 -j myFileset
[2014-10-11 02:39:11] Prepare restore ...
[2014-10-11 02:39:17] Scan inodes ...
[2014-10-11 02:39:19] 196928 inodes have been scanned. 50% of total.
[2014-10-11 02:39:22] 393856 inodes have been scanned. 100% of total.
[2014-10-11 02:39:22] Construct operation list ...
[2014-10-11 02:39:22] 0 operations have been added to list.
[2014-10-11 02:39:22] 1 operations have been added to list.
[2014-10-11 02:39:22] Restore files ...
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     893440        5242880    17.041015625%
[I] 4030 of 266880 inodes used: 1.510042%.
[I] Loaded policy rules from /GPFS/myFileset/RestoreTemp17773/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:39:22 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -j myFileset -o
myfsetsnap01 -t /GPFS/myFileset/RestoreTemp17773 -B 100  -v 1   -p 3 -c 17773'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/myFileset/RestoreTemp17773/opList...
[I] 2014-10-10@23:39:22.790 Executing file list: /GPFS/myFileset/RestoreTemp17773/opList. 1 files
dispatched.
[I] A total of 1 PDRs from filelist /GPFS/myFileset/RestoreTemp17773/opList have been processed; 0 'skipped'
records and/or errors.
[I] A total of 1 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
   0 'skipped' files and/or errors.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     893440        5242880    17.041015625%
[I] 4029 of 266880 inodes used: 1.509667%.
[I] Loaded policy rules from /GPFS/myFileset/RestoreTemp17773/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:39:22 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -j myFileset -o
myfsetsnap01 -t /GPFS/myFileset/RestoreTemp17773 -B 100  -v 1   -p 51 -c 17773'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/myFileset/RestoreTemp17773/opList.copy...
```

```
[I] 2014-10-10@23:39:23.352 Executing file list: /GPFS/myFileset/RestoreTemp17773/opList.copy. 0 files
dispatched.
[I] A total of 0 PDRs from filelist /GPFS/myFileset/RestoreTemp17773/opList.copy have been processed; 0
'skipped' records and/or errors.
[I] A total of 0 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
    0 'skipped' files and/or errors.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                        66048        5242880     1.259765625%
silver                      66048        5242880     1.259765625%
system                     893440        5242880    17.041015625%
[I] 4029 of 266880 inodes used: 1.509667%.
[I] Loaded policy rules from /GPFS/myFileset/RestoreTemp17773/policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-10@23:39:23 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'all'  EXEC '/usr/lpp/mmfs/bin/tsrestorefileset' OPTS ' -d /dev/GPFS -j myFileset -o
myfsetsnap01 -t /GPFS/myFileset/RestoreTemp17773 -B 100  -v 1   -p 52 -c 17773'
       RULE 'restore' LIST 'all' DIRECTORIES_PLUS
       SHOW('d') WHERE MODE LIKE 'd%'
[I] Executing filelist: /GPFS/myFileset/RestoreTemp17773/opList.copy.delta...
[I] 2014-10-10@23:39:23.904 Executing file list: /GPFS/myFileset/RestoreTemp17773/opList.copy.delta. 0 files
dispatched.  \..[I] 2014-10-10@23:39:23.925 Executing file list:
/GPFS/myFileset/RestoreTemp17773/opList.copy.delta. 0 files dispatched.
[I] A total of 0 PDRs from filelist /GPFS/myFileset/RestoreTemp17773/opList.copy.delta have been processed;
0 'skipped' records and/or errors.
[I] A total of 0 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
    0 'skipped' files and/or errors.
[2014-10-11 02:39:23] Restore completed.
[2014-10-11 02:39:23] Clean up.#
```

Note: The **mmrestorefs** command can take a long time to process, depending on the fileset size. It is possible to define the number of threads, files per thread, and nodes for this operation from the defaults of 24 threads, 60 files per thread, and all nodes.

5.3.3 Snapshot at file level

IBM Spectrum Scale allows snapshots at file level. They are referred to as *file clones*. Creating a file clone from a regular file for writing is a two-step process that uses the **mmclone** command with the snap and copy keywords:

1. Issue the **mmclone snap** command to create a read-only snapshot of the file to be cloned. This read-only snapshot becomes known as the *clone parent*. This step creates the actual snapshot of the file. If no write access is required for the clone, there is no need to perform the second step.
2. Issue the **mmclone copy** command to create a writable clone from a clone parent.

Example 5-27 shows how to use the **mmclone** command to create a read-only file clone. The example also uses the cloned parent to create a writable copy to write to it.

Example 5-27 Creating a file clone from a file

```
To create a the read only snapshot of the file:
# mmclone snap 100M.raw 100M.raw_snap

Once a read only source is available to create a writable clone from the snapshot
file use the following:
# mmclone copy 100M.raw_snap 100M.raw_cloned
```

In addition to using a regular file as source, Spectrum Scale can also create a file clone from a file in a global or fileset snapshot. It can be done by using the **mmclone** command as shown in Example 5-28.

Example 5-28 Creating a file clone from a file in a snapshot

```
# mmclone copy /GPFS/myFileset/snapdir/myfsetsnap01/myfile1 myfile1_from_snap
```

Note: Do not use snapshots as protection against media failures or backup solution.

5.4 Storage pools

Physically, a storage pool is a collection of disks or Redundant Array of Independent Disks (RAID) arrays. Storage pools also allow you to group multiple storage systems within a file system.

By using storage pools, you can create tiers of storage by grouping storage devices based on performance, locality, or reliability characteristics. For example, one pool could be an enterprise class storage system that hosts high-performance Fibre Channel disks and another pool might consist of numerous disk controllers that host a large set of economical SATA disks.

There are two types of storage pools in Spectrum Scale, internal storage pools and external storage pools. Internal storage pools are managed within Spectrum Scale. External storage pools are managed by an external application, such as Tivoli Storage Manager. For external storage pools, Spectrum Scale provides tools that allow you to define an interface that your external storage manager uses to access your data. Spectrum Scale does not manage the data placed in external storage pools. Instead, Spectrum Scale manages the movement of data to and from external storage pools. Storage pools allow you to perform complex operations such as moving, mirroring, or deleting files across multiple storage devices, providing storage virtualization and a single management context.

Internal Spectrum Scale storage pools are meant for managing online storage resources. External storage pools are intended for use as near-line storage and for archival and backup operations. However, both types of storage pools provide you with a method to partition file system storage for considerations such as:

- Improved price-performance by matching the cost of storage to the value of the data
- Improved performance by:
  - Reducing the contention for premium storage
  - Reducing the impact of slower devices
  - Allowing you to retrieve archived data when needed
- Improved reliability by providing for:
  - Replication based on need
  - Better failure containment
  - Creation of new storage pools as needed

5.4.1 Internal storage pools

The internal Spectrum Scale storage pool to which a disk belongs is specified as an attribute of the disk in the Spectrum Scale cluster. You specify the disk attributes as a field in each disk descriptor when you create the file system or when adding disks to an existing file system.

Spectrum Scale allows a maximum of eight internal storage pools per file system. One of these storage pools is the required system storage pool. The other seven internal storage pools are optional user storage pools.

System storage pool

The system storage pool contains file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks, extended attributes, and so on. The system storage pool can also contain user data. There is only one system storage pool per file system, and it is automatically created when the file system is created.

System storage note: Use highly reliable disks and replication for the system storage pool because it contains system metadata.

System.log storage pool

By default, the file system recovery log is stored in the system storage pool with file system metadata. The file system recovery log can also be placed in a dedicated pool that is called the system.log pool. This storage pool must be created explicitly. It is highly recommended to only use storage that is as fast or even faster than what is used for the system storage pool. This recommendation is because of the high number of small synchronous data updates made to the recovery log. The block size for the system.log pool must be the same as the block size of the system pool.

Note: There can be at most one system.log storage pool per file system.

Figure 5-2 shows a cluster with only user-defined pool.

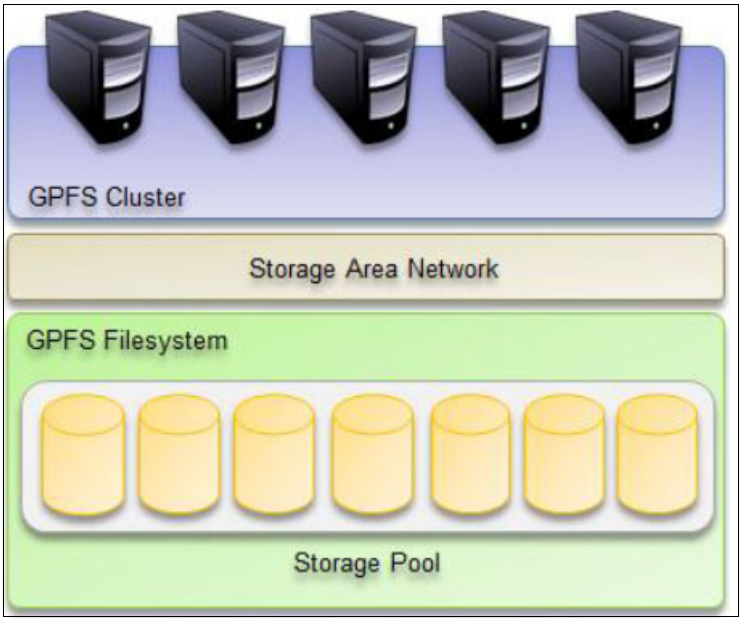


Figure 5-2 Spectrum Scale cluster with one user-defined pool

User-defined storage pool

A user storage pool only contains the blocks of data (user data, for example) that make up a user file. Spectrum Scale stores the data that describes the files, called *file metadata*, separately from the actual file data in the system storage pool. You can create one or more

user storage pools, and then create policy rules to indicate where the data blocks for a file should be stored.

Note: There is a maximum of eight (system + seven user defined) internal pools per file system.

Figure 5-3 shows a cluster with two user-defined pools.

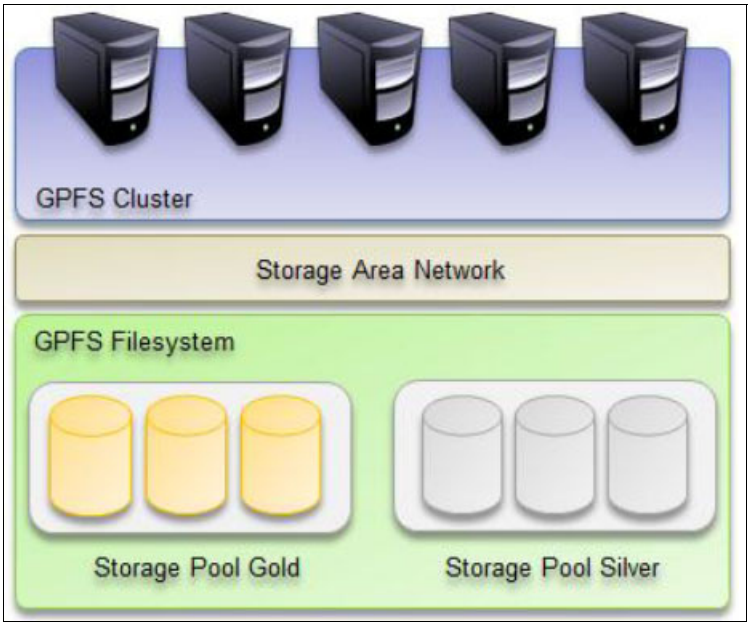


Figure 5-3 Spectrum Scale cluster with two user-defined pools

Note: The storage pool of an NSD is defined at creation time. Metadata can only be stored in the system pool. The system pool has to always exist and is not being shown in Figure 5-3.

It is possible to show the pools of a mounted Spectrum Scale file system with the **mmlspool** command and the name of the device. Example 5-29 shows how the pool looks with two user-defined pools.

Example 5-29 Showing pools with the mmlspool command

```
# mmlspool shared
Storage pools in file system at '/gpfs/shared':
Name                    Id   BlkSize Data Meta Total Data in (KB)   Free Data in (KB)   Total Meta in (KB)    Free Meta in (KB)
system                   0    256 KB  yes  yes       67108864       66069504 ( 98%)       67108864       66070016 ( 98%)
GOLD                 65537    256 KB  yes   no       67108864       67042816 (100%)              0              0 (  0%)
SILVER               65538    256 KB  yes   no       67108864       67042816 (100%)              0              0 (  0%)
```

5.4.2 External storage pools

When you initially create a file, IBM Spectrum Scale assigns that file to an internal storage pool. Internal storage pools support various types of online storage. To move data from online storage to offline or near-line storage, you can use external storage pools. External storage pools use a flexible interface driven by Spectrum Scale policy rules that simplify data migration to and from other types of storage such as tape storage.

You can define multiple external storage pools at any time by using IBM Spectrum Scale policy rules. To move data to an external storage pool, the Spectrum Scale policy engine evaluates the rules that determine which files qualify for transfer to the external pool. From that information, Spectrum Scale provides a list of candidate files and executes the script that is specified in the rule that defines the external pool. That executable script is the interface to the external application, such as Tivoli Storage Manager, that does the actual migration of data into an external pool.

Using the external pool interface, Spectrum Scale gives you the ability to manage information by allowing you to:

- Move files and their extended attributes onto low-cost near-line or auxiliary storage when demand for the files diminishes.
- Recall the files, with all of their previous access information, onto online storage whenever the files are needed.

The external storage manager is responsible for moving files from Spectrum Scale and returning them upon the request of an application accessing the file system. Therefore, when you are using external storage pools, you must use an external file management application. The external application is responsible for maintaining the file once it has left the Spectrum Scale file system.

For example, with Tivoli Storage Manager, Spectrum Scale policy rules create a list of files that are eligible for migration. IBM Spectrum Scale hands that list to Tivoli Storage Manager, which migrates the files to tape and creates a reference file in the file system that has pointers to the tape image. When a file is requested, it is automatically retrieved from the external storage pool and placed back in an internal storage pool. As an alternative, you can use a Spectrum Scale policy rule to retrieve the data in advance of a user request.

The number of external storage pools is only limited by the capabilities of your external application. Spectrum Scale allows you to define external storage pools at any time by writing a policy that defines the pool and makes that location known to Spectrum Scale.

The hierarchical storage management (HSM) function is implemented differently in Tivoli Storage Manager for Windows operating systems and UNIX. In the Windows systems environment, it is called HSM for Windows; in the UNIX part it is called Tivoli Storage Manager for Space Management. There are major differences in the way they are implemented in their respective supported platforms. Figure 5-4 on page 264 shows the typical scenario of Tivoli Storage Manager for Space Management.

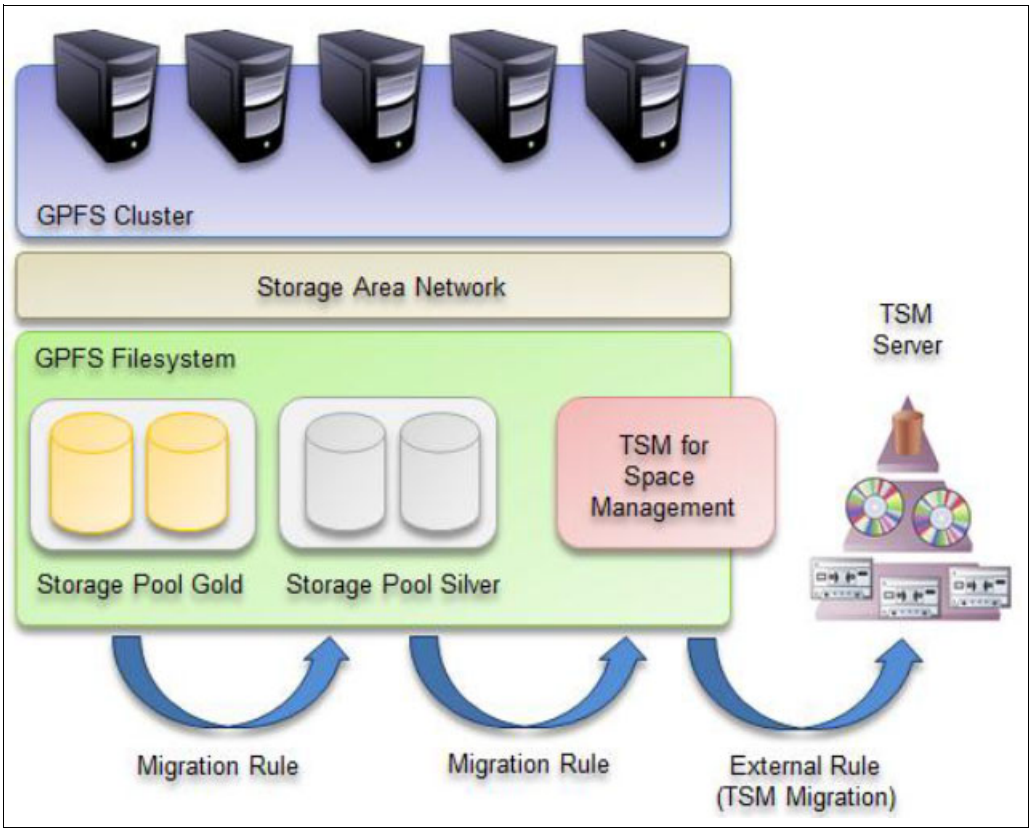


Figure 5-4 Tivoli Storage Manager for Space Management

For more information about HSM and Space Management and how to configure it, refer to the IBM Redbooks publication *IBM Tivoli Storage Manager as a Data Protection Solution*, SG24-8134:

`http://www.redbooks.ibm.com/abstracts/sg248134.html?Open`

See also the Tivoli Field Guide *TSM for Space Management for UNIX –GPFS Integration:*

`http://www-01.ibm.com/support/docview.wss?uid=swg27028178`

If the objective is only to use tapes as tiers on Spectrum Scale, IBM offers *Linear Tape File System Enterprise Edition* (LTFS EE) product. LTFS EE for the IBM TS4500, IBM System Storage TS3500, and IBM System Storage TS3310 Tape Libraries provide seamless integration of LTFS with Spectrum Scale by creating an LTFS tape tier. You can run any application that is designed for disk files on tape by using LTFS EE.

The use of LTFS EE to replace disks with tape in Tier 2 and Tier 3 storage can improve data access over other storage solutions because it improves efficiency and streamlines management for files on tape. LTFS EE simplifies the use of tape by making it not apparent to the user and manageable by the administrator under a single infrastructure. Figure 5-5 on page 265 shows the typical LTFS EE setup.

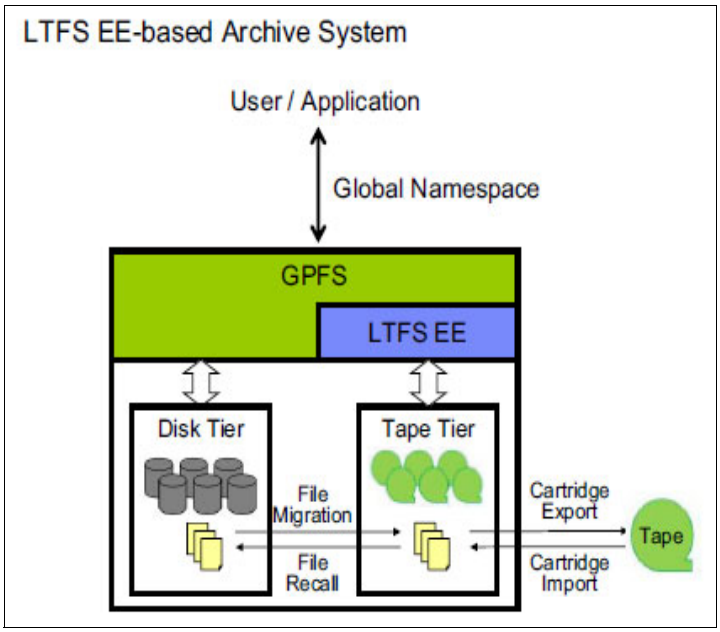


Figure 5-5 High-level overview of LTFS EE archive solution

For more information about LTFS EE and how to configure it, see the IBM Redbooks publication *IBM Linear Tape File System Enterprise Edition V1.1.1.2: Installation and Configuration Guide*, SG24-8143:

`http://www.redbooks.ibm.com/redpieces/abstracts/sg248143.html`

Usage of External pools on Spectrum Scale is not limited to the preceding products. Spectrum Scale offers flexibility to run with other products or even custom made solutions.

5.5 Immutability and appendOnly attributes

To prevent files from being changed or deleted unexpectedly, Spectrum Scale provides *immutability* and *appendOnly* restrictions.

You can apply *immutability* and *appendOnly* restrictions either to individual files within a fileset or to a directory. When using *immutability* or *appendOnly* restrictions, have in mind the following:

- An `immutable` file cannot be changed or renamed. An `appendOnly` file allows append operations, but not delete, modify, or rename operations.
- An `immutable` directory cannot be deleted or renamed, and files cannot be added or deleted under such a directory. An `appendOnly` directory allows new files or subdirectories to be created with 0-byte length; all such new created files and subdirectories are marked as `appendOnly` automatically.
- The **`immutable`** flag and the **`appendOnly`** flag can be set independently. If both `immutability` and `appendOnly` are set on a file, `immutability` restrictions are in effect.
- Before an `immutable` or `appendOnly` file can be deleted, you must change it to `mutable` or set `appendOnly` to no.
- Storage pool assignment of an `immutable` or `appendOnly` file can be changed; an `immutable` or `appendOnly` file is allowed to transfer from one storage pool to another.

To set or unset **Immutability** and **appendOnly** attributes, the **mmchattr** command must be used. Example 5-30 shows how to change a file from mutable to immutable.

Example 5-30 Setting and unsetting immutable and appendOnly attributes in a file

```
To make myfile immutable:
# mmchattr -i yes myfile

To make myfile mutable:
# mmchattr -i no myfile

To set appendOnly on myfile:
# mmchattr -a yes myfile

To unset appendOnly on myfile:
# mmchattr -a no myfile
```

Note: Setting or unsetting **Immutability** and **appendOnly** attributes in a directory is done in the same way as with a file.

To display if a file or directory is *immutable* or *appendOnly*, the **mmlsattr** command must be used. Example 5-31 shows the attributes for one file.

Example 5-31 Listing attributes for a file

```
# mmlsattr -L myfile
file name:            myfile
metadata replication: 1 max 3
data replication:     1 max 3
immutable:            yes
appendOnly:           yes
flags:
storage pool name:    system
fileset name:         myFileset
snapshot name:
creation time:        Mon Oct 13 16:21:47 2014
Windows attributes:   ARCHIVE READONLY
Encrypted:            no
```

Table 5-1 compares the operations that are allowed on `immutable` and `appendOnly` files.

Table 5-1 Effect of file operations on immutable and appendOnly files

Operation	immutable	appendOnly
Add, delete, modify, or rename	No	No
Append	No	Yes
Change ownership, mode, or acl	No	No
Change atime, mtime, or ctime	Yes	Yes
Add, delete, or modify extended attributes	Disallowed by external methods such as **setfattr** Allowed internally for **dmapi**, **directio**, and others	Same as for immutable

Operation	immutable	appendOnly
Create a file under an immutable or appendOnly directory	No	Yes, 0-byte length only
Rename or delete a file under an immutable or appendOnly directory	No	No
Modify a mutable file under an immutable directory	Yes	Not applicable
Set an immutable file back to mutable	Yes	Not applicable
Set an appendOnly file back to a non-appendOnly state	Not applicable	Yes

5.6 Quotas

The IBM Spectrum Scale quota system helps you to control the allocation of files and data blocks in a file system.

Spectrum Scale quotas can be defined for:

- Individual users
- Groups of users
- Individual filesets

Note: Windows operating system nodes may be included in clusters that use Spectrum Scale quotas. Those nodes are enforced to comply with the quotas. However, Windows operating system nodes do not support the quota commands and are not able to run such commands in the cluster.

To have quota enabled on a Spectrum Scale file system, the **`-Q yes`** parameter must be enabled, either at creation with the **`mmcrfs`** command or afterward with the **`mmchfs`** command.

Note: If you are planning to use quotas on a file system level only, it can be done online with the **`mmchfs`** command. If you are planning to use quotas at a fileset level, be aware that they cannot be activated online.

When using quotas at a fileset level (**`--perfileset-quota`** option), you might want to enable also the **`--filesetdf`** option. In that way, you get the space free based on the quota for that fileset.

Note: **`df`** needs to have the full path of the junction to show the quota at fileset level.

5.6.1 Enabling and editing quotas

This section shows how to enable quotas on the file system. The following examples show how to define a soft block quota of 5 GB, and hard block quota of 10 GB in a previously created fileset. There are no limits on number of inodes (allocation of files).

Example 5-32 shows how to enable the quotas on the file system if it was not done already.

Example 5-32 Setting block quotas in a fileset

```
# mmquotaon -v GPFS
mmquotaon on GPFS
```

Example 5-33 shows how to list current quotas for the fileset.

Example 5-33 Listing current quotas for the fileset

```
# mmlsquota -j myFileset GPFS
                         Block Limits         |              File Limits
Filesystem type KB quota limit in_doubt grace | files quota limit in_doubt grace  Remarks
GPFS       FILESET     no limits
```

Example 5-34 shows how to edit the quota for the fileset.

Example 5-34 Editing the quota for the fileset

```
# mmedquota -j GPFS:myFileset
```

By running the command in Example 5-34, your default editor opens with the contents of the quota for the fileset. See Example 5-35.

Example 5-35 Default editor opens with the contents of the quota for the fileset

```
*** Edit quota limits for FILESET myFileset
NOTE: block limits will be rounded up to the next multiple of the block size.
      block units may be: K, M, G, T or P, inode units may be: K, M or G.
GPFS: blocks in use: 102400K, limits (soft = 0K, hard = 0K)
        inodes in use: 7, limits (soft = 0, hard = 0)
```

Modify the line that contains the “GPFS: blocks” so it has soft quota of 5 G and hard quota of 10 G. And save the file.

Example 5-36 lists the quotas for the fileset after the change.

Example 5-36 Listing the quotas for the fileset after the change

```
# mmlsquota -j myFileset GPFS
                         Block Limits                                                |
            File Limits
Filesystem type    KB     quota   limit    in_doubt grace| files quota limit in_doubt grace  Remarks
GPFS       FILESET 102400 5242880 10485760 0        none | 7     0     0     0               none
```

Note: To be able to use quotas, the file system needs to have the **-Q** yes parameter configured. It can be done at creation time with the **mmcrfs** command or once created with the **mmchfs** command.

If for whatever reason your quota accounting information is outdated, and you get a message informing you of that, update the quota accounting information with the **mmcheckquota** command as shown in Example 5-37 on page 269.

Example 5-37 Updating the quota accounting information

```
# mmcheckquota GPFS
GPFS: Start quota check
   6 % complete on Mon Oct 13 17:11:24 2014
  13 % complete on Mon Oct 13 17:11:25 2014
  40 % complete on Mon Oct 13 17:11:26 2014
  47 % complete on Mon Oct 13 17:11:27 2014
  55 % complete on Mon Oct 13 17:11:28 2014
  80 % complete on Mon Oct 13 17:11:29 2014
  87 % complete on Mon Oct 13 17:11:30 2014
  95 % complete on Mon Oct 13 17:11:31 2014
 100 % complete on Mon Oct 13 17:11:32 2014
Finished scanning the inodes for GPFS.
Merging results from scan.
```

You can also use the **mmsetquota** command to apply quota limits. For more information about the **mmsetquota** command, see *IBM Spectrum Scale 4.1: Administration and Programming Reference,* SA23-1452-00.

5.6.2 Creating quota reports

To create quota reports, use the **mmrepquota** command. The **mmrepquota** command reports file system usage and quota information for a user, group, or fileset.

This command cannot be run from a Windows operating system node. If **-g**, **-j**, or **-u** flags are not specified, then user, group, and fileset quotas are listed. If **-a** flag is not specified, the device must be the last parameter entered.

For each file system in the cluster, the **mmrepquota** command displays:

1. Block limits:
   - Quota type (USR, GRP, or FILESET)
   - Current usage
   - Soft limit
   - Hard limit
   - Space in doubt
   - Grace period
2. File limits:
   - Current number of files
   - Soft limit
   - Hard limit
   - Files in doubt
   - Grace period

   Note: In cases where small files do not have an additional block allocated for them, quota usage may show less space usage than expected.

3. Entry Type
   - **default on**

     Default quotas are enabled for this file system.
   - **default off**

     Default quotas are not enabled for this file system.

– **e**

  Explicit quota limits have been set by using the **mmedquota** command.

– **d_fsys**

  The quota limits are the default file system values set by using the **mmdefedquota** command.

– **d_fset**

  The quota limits are the default fileset-level values set by using the **mmdefedquota** command.

– **i**

  Default quotas were not enabled when this initial entry was established. Initial quota limits have a value of zero indicating no limit.

For example, to report on user quotas for file system fs2 and display a header line, issue the command as shown in Example 5-38.

Example 5-38 The mmrepquota command

```
$ mmrepquota -u -v fs2
The system displays information similar to:
*** Report for USR quotas on fs2
Block Limits | File Limits
in in entry
Name type KB quota limit doubt grace |files quota limit doubt grace Type
root USR 8 0 0 0 none | 1 0 0 0 none default on
user2 USR 2016 256 512 0 6days | 7 10 20 0 none d_fsys
user3 USR 104 256 512 0 none | 1 10 20 0 none d_fsys
user4 USR 0 256 512 0 none | 0 10 20 0 none d_fsys
user5 USR 368 256 512 0 23hours | 5 4 10 0 23hours d_fsys
user6 USR 0 256 512 0 none | 0 10 20 0 none d_fsys
user7 USR 1024 1024 5120 4096 none | 1 0 0 19 none e
```

5.7 Policies and rules

Policies and the rules that they contain are used to assign files to specific storage pools. A storage pool typically contains a set of volumes that provide a specific quality of service for a specific use.

5.7.1 Policies

IBM Spectrum Scale supports the following policies:

- File placement policies are used to automatically place newly created files in a specific storage pool.
- File management policies are used to manage files during their lifecycle by moving them to another storage pool, moving them to near-line storage, copying them to archival storage, changing their replication status, or deleting them.

A *policy* is a set of rules that describes the lifecycle of user data based on the file's attributes. Each *rule* defines an operation or definition, such as migrate to a pool and replicate the file.

The three uses for rules are as follows:

- Initial file placement
- File management
- Restoring file data

When a file is created or restored, the placement policy determines the location of the file's data and assigns the file to a storage pool. All data written to that file is placed in the assigned storage pool.

The placement policy defines the initial placement of newly created files; the rules for placement of restored data must be installed into Spectrum Scale with the **mmchpolicy** command. If a Spectrum Scale file system does not have a placement policy installed, all the data will be stored in the system storage pool. Only one placement policy can be installed at a time. If you switch from one placement policy to another, or make changes to a placement policy, that action has no effect on existing files. However, newly created files are always placed according to the currently installed placement policy.

Characteristics of a policy are as follows:

- A policy can contain any number of rules.
- A policy file is limited to a size of 1 MB.

5.7.2 Rules

A *rule* is an SQL-like statement that tells Spectrum Scale what to do with the data for a file in a specific storage pool, if the file meets specific criteria. A rule can apply to any file being created or only to files being created within a specific file set or group of file sets.

Rules specify conditions that, when true, cause the rule to be applied. Conditions that cause Spectrum Scale to apply a rule are as follows:

- Date and time when the rule is evaluated, that is, the current date and time
- Date and time when the file was last accessed
- Date and time when the file was last modified
- File set name
- File name or extension
- File size
- User ID and group ID

5.7.3 Example of polices with internal pools only

In this example, we define a Spectrum Scale file system with three internal pools:

- *system* pool, with solid-state drives (SSDs) that hold data and metadata.
- *gold* pool, with tier 1 storage area network (SAN) disks that hold data.
- *silver* pool, with tier 3 SAN disks that hold data.

The file system has the following set of policies:

- By default, new files go to *system* pool.
- The *clean_system* policy moves files to *gold* pool when a certain threshold of utilization of the *system* pool is reached.
- The *clean_gold* policy moves files every Sunday to *silver* pool that were not accessed in the last 15 days.

- The *delete_silver* policy deletes files every first of the month from *silver* pool that their access time is larger than 365 days.
- The *upgrade_to_gold* policy moves files every day from *silver* pool to *gold* pool if their access time is less than two days, file is larger than 9999 KB, and is not owned by *root* user.

Note: For threshold rules such as *clean_system*, a callback is added to work automatically. Add the following callback from any node and threshold rules are enforced automatically, otherwise they are enforced manually:

```
# mmaddcallback MIGRATION --command /usr/lpp/mmfs/bin/mmstartpolicy --event
lowDiskSpace,noDiskSpace --parms "%eventName %fsName"
```

Figure 5-6 shows the intended policies flows and relationships with the pools.

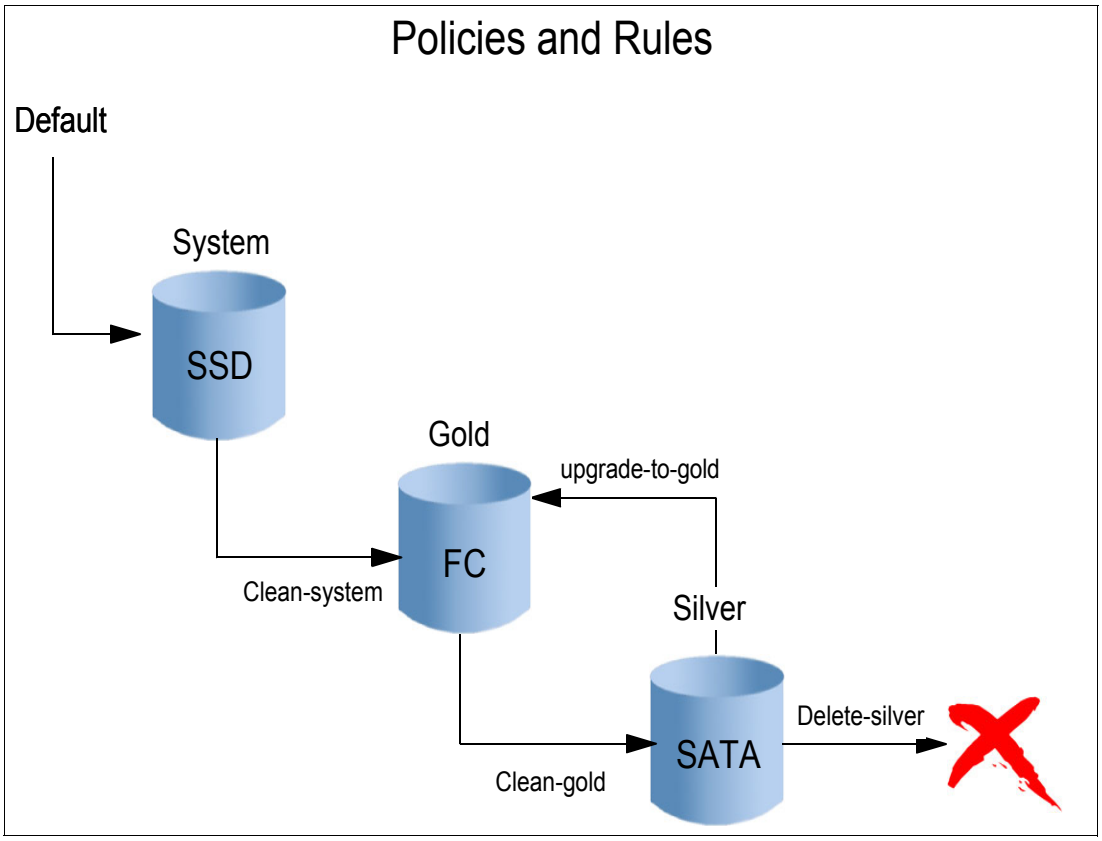


Figure 5-6 ILM example rules and pools relationship

This scenario has the following limitations:

- Files are deleted at some point on their life. So a system to back up those might be needed depending on the type of files and system.
- It has no consideration on multiple copies into different storage subsystems. All data is only replicated once. In a production scenario that might not be the wanted setup.
- Files never get upgraded pool with SSD. In this scenario, it is assumed that tier 1 SAN has IBM Spectrum Virtualize with IBM Easy Tier® functionality.
- Only internal pools are used to move around data. External pools and filesets can be used as well. So data could end on tape instead of being deleted.

- The *system* pool is used for data and metadata. It might be more realistic to use it only for metadata.
- Only files are evaluated. No `DIRECTORIES_PLUS` is used.
- The *gold* and *silver* pools can be filled up and Spectrum Scale will not react to that.
- The scenario is only a showcase of Spectrum Scale ILM capabilities, and is not intended to be a guide to “fit all” scenarios.

Several rule types exist:

- Placement policies, evaluated at file creation (*clean_system*).
- Migration policies, evaluated periodically (*clean_gold*, *upgrade_to_gold*).
- Deletion policies, evaluated periodically (*delete_silver*).

To create a policy for your use case, create a text file for your policy with the following guidelines:

- A policy must contain at least one rule.
- The last placement rule of a policy rule list must be as though no other placement rules apply to a file; the file is assigned to a default pool.

See the policy that fits the use case defined as shown in Example 5-39.

Example 5-39 Policy file for the example use case

```
RULE 'clean_system' MIGRATE FROM POOL 'system' THRESHOLD(85,40)
        WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'

RULE 'clean_gold'
        MIGRATE FROM POOL 'gold'
        WEIGHT(KB_ALLOCATED)
        TO POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '15' DAYS)

RULE 'delete_silver' WHEN (DAYOFWEEK(CURRENT_DATE) NOT IN (7,1)) /* Does NOT move data on weekends */
        DELETE FROM POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '365' DAYS)

RULE 'upgrade_to_gold'
    MIGRATE FROM POOL 'silver'
    WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'
    WHERE (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '2' DAYS AND KB_ALLOCATED > 9999 AND NOT USER_ID=0)

RULE 'default' SET POOL 'system'
```

The rule *clean_system* is evaluated periodically. In our example, we have a callback so it is automatically triggered when the threshold is reached. When the *system* pool reaches the high occupancy threshold of 85%, the Spectrum Scale file system ILM moves files from *system* pool to *gold* pool until the low occupancy threshold of 40% on *system* pool is reached. The order on which the files are migrated is based on the weight that in our case is based on file size.

The rule *clean_gold* is evaluated periodically. In our example, the **`mmapplypolicy`** command will be run as cronjob every 12 hours. It moves files from *gold* pool to *silver* pool if their access time (*atime* on UNIX) is older than 15 days.

The rule *delete_silver* is evaluated periodically. In our example, **mmapplypolicy** command will be run as cronjob every 12 hours. It deletes files from *silver* pool if their access time (atime on UNIX) is older than 365 days and is not weekend.

The rule *upgrade_to_gold* is evaluated periodically. In our example, **mmapplypolicy** command will be run as cronjob every 12 hours. It moves files from *silver* pool to *gold* pool if the access time (atime on UNIX) is younger than two days, the size of the file is larger than 9999 KB, and is not owned by *root* user.

The *default* rule is evaluated every time that a file is created. That means the default rule is automatically enforced without any cron or manual run every certain period of time. When a file is created it allocates into the *system* pool.

Note: Evaluating a rule does not mean that the rule is executed. Only the files that hit the policy will then be moved accordingly. However, files would only be migrating when the rules are being evaluated, not in between evaluations, with the exception of the use of callbacks. In this setup, *clean_system* is evaluated “all” the time.

The **WEIGHT** parameter establishes an order on the matching files. The **WEIGHT** parameter specifies an SQL expression with a numeric value that can be converted to a double precision floating point number. The expression can refer to any of the file attributes and may include any constants and any of the available SQL operators or built-in functions.

After the policy file is created, is time to install the policy. Run the **mmchpolicy** command to validate and install the policy and the **mmlspolicy** command to list the installed policies as it is shown in Example 5-40, Example 5-41, and Example 5-43 on page 275.

The policy file is called GPFS_rules and contains the text as shown on Example 5-39 on page 273. First, check that the policy has no syntax errors, as shown in Example 5-40.

Example 5-40 Installing a policy into the Spectrum Scale file system: Checking that the policy has no syntax errors

```
# mmchpolicy GPFS GPFS_rules -I test
Validated policy `GPFS_rules': Parsed 5 policy rules.
```

As shown in Example 5-41, once it is validated, it can be installed into the Spectrum Scale file system.

Example 5-41 Once validated, it can be installed into the Spectrum Scale file system

```
# mmchpolicy GPFS GPFS_rules -I yes
Validated policy `GPFS_rules': Parsed 5 policy rules.

Policy `GPFS_rules' installed and broadcast to all nodes.
```

Once installed, list the policies that are known to the Spectrum Scale file system, as shown in Example 5-42.

Example 5-42 Listing the policies known to the Spectrum Scale file system

```
# mmlspolicy GPFS -L
RULE 'clean_system' MIGRATE FROM POOL 'system' THRESHOLD(85,40)
        WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'

RULE 'clean_gold'
        MIGRATE FROM POOL 'gold'
```

```
WEIGHT(KB_ALLOCATED)
        TO POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '15' DAYS)

RULE 'delete_silver' WHEN (DAYOFWEEK(CURRENT_DATE) NOT IN (7,1)) /* Does NOT move data on weekends */
        DELETE FROM POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '365' DAYS)

RULE 'upgrade_to_gold'
    MIGRATE FROM POOL 'silver'
    WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'
    WHERE (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '2' DAYS AND KB_ALLOCATED > 9999 AND NOT USER_ID=0)

RULE 'default' SET POOL 'system'
```

Note: To change a policy, modify the file with all the rules that are wanted and install the policy as in Example 5-40 on page 274.

Once the rules are installed into the Spectrum Scale file system, it is possible to manually enforce the rules without waiting for a cron job or callback to be triggered. Example 5-43 shows how to use the **mmapplypolicy** command to test the policy.

Example 5-43 Running mmapplypolicy to test the policy attributes and parameters

```
# mmapplypolicy GPFS -I test
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                      3314176        5242880    63.212890625%
silver                      66048        5242880     1.259765625%
system                    1945088        5242880    37.099609375%
[I] 4034 of 266880 inodes used: 1.511541%.
[I] Loaded policy rules from /var/mmfs/tmp/cmdTmpDir.mmapplypolicy.21376/tspolicyFile.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-13@21:51:47 UTC
Parsed 5 policy rules.
     RULE 'clean_system' MIGRATE FROM POOL 'system' THRESHOLD(85,40)
        WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'

RULE 'clean_gold'
        MIGRATE FROM POOL 'gold'
        WEIGHT(KB_ALLOCATED)
        TO POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '1' DAYS)

RULE 'delete_silver' WHEN (DAYOFWEEK(CURRENT_DATE) NOT IN (7,1)) /* Does NOT move data on weekends */
        DELETE FROM POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '10' DAYS)

RULE 'upgrade_to_gold'
    MIGRATE FROM POOL 'silver'
    WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'
    WHERE (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '5' DAYS AND KB_ALLOCATED > 9999 AND NOT USER_ID=0)

RULE 'default' SET POOL 'system'
[I] 2014-10-13@21:51:47.876 Directory entries scanned: 24.
[I] Directories scan: 22 files, 2 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@21:51:47.879 Sorting 24 file list records.
```

```
[I] Inodes scan: 22 files, 2 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@21:51:47.922 Policy evaluation. 24 files scanned.
[I] 2014-10-13@21:51:47.930 Sorting 3 candidate file list records.
[I] 2014-10-13@21:51:47.931 Choosing candidate files. 3 records scanned.
[I] Summary of Rule Applicability and File Choices:
 Rule#    Hit_Cnt     KB_Hit     Chosen  KB_Chosen     KB_IllRule
     0          0          0          0          0          0RULE 'clean_system' MIGRATE FROM POOL
'system' THRESHOLD(85.000000,40.000000) WEIGHT(.) TO POOL 'gold'
     1          3          0          3          0          0RULE 'clean_gold' MIGRATE FROM POOL 'gold'
WEIGHT(.) TO POOL 'silver' WHERE(.)
     2          0          0          0          0          0RULE 'delete_silver' WHEN(.) DELETE FROM
POOL 'silver' WHERE(.)
     3          0          0          0          0          0RULE 'upgrade_to_gold' MIGRATE FROM POOL
'silver' WEIGHT(.) TO POOL 'gold' WHERE(.)

[I] Filesystem objects with no applicable rules: 21.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to migrate 0KB: 3 of 3 candidates;
Predicted Data Pool Utilization in KB and %:
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                      3314176        5242880    63.212890625%
silver                      66048        5242880     1.259765625%
system                    1945088        5242880    37.099609375%
```

The dry run shows that three files hit *clean_gold* rule. Example 5-44 shows a cron job running the **mmapplypolicy** command every 12 hours.

Example 5-44 Running mmapplypolicy to apply policy

```
# mmapplypolicy GPFS -I yes
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name             KB_Occupied       KB_Total Percent_Occupied
gold                      3314176        5242880    63.212890625%
silver                      66048        5242880     1.259765625%
system                    1945088        5242880    37.099609375%
[I] 4034 of 266880 inodes used: 1.511541%.
[I] Loaded policy rules from /var/mmfs/tmp/cmdTmpDir.mmapplypolicy.21508/tspolicyFile.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-13@21:56:37 UTC
Parsed 5 policy rules.
     RULE 'clean_system' MIGRATE FROM POOL 'system' THRESHOLD(85,40)
        WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'

RULE 'clean_gold'
        MIGRATE FROM POOL 'gold'
        WEIGHT(KB_ALLOCATED)
        TO POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '1' DAYS)

RULE 'delete_silver' WHEN (DAYOFWEEK(CURRENT_DATE) NOT IN (7,1)) /* Does NOT move data on weekends */
        DELETE FROM POOL 'silver'
        WHERE (CURRENT_TIMESTAMP - ACCESS_TIME > INTERVAL '10' DAYS)

RULE 'upgrade_to_gold'
    MIGRATE FROM POOL 'silver'
    WEIGHT(KB_ALLOCATED)
        TO POOL 'gold'
    WHERE (CURRENT_TIMESTAMP - ACCESS_TIME < INTERVAL '5' DAYS AND KB_ALLOCATED > 9999 AND NOT USER_ID=0)
```

```
RULE 'default' SET POOL 'system'
[I] 2014-10-13@21:56:37.675 Directory entries scanned: 24.
[I] Directories scan: 22 files, 2 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@21:56:37.678 Sorting 24 file list records.
[I] Inodes scan: 22 files, 2 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@21:56:37.711 Policy evaluation. 24 files scanned.
[I] 2014-10-13@21:56:37.717 Sorting 3 candidate file list records.
[I] 2014-10-13@21:56:37.718 Choosing candidate files. 3 records scanned.
[I] Summary of Rule Applicability and File Choices:
 Rule#    Hit_Cnt     KB_Hit     Chosen  KB_Chosen     KB_IllRule
     0          0          0          0          0          0RULE 'clean_system' MIGRATE FROM POOL
'system' THRESHOLD(85.000000,40.000000) WEIGHT(.) TO POOL 'gold'
     1          3          0          3          0          0RULE 'clean_gold' MIGRATE FROM POOL 'gold'
WEIGHT(.) TO POOL 'silver' WHERE(.)
     2          0          0          0          0          0RULE 'delete_silver' WHEN(.) DELETE FROM
POOL 'silver' WHERE(.)
     3          0          0          0          0          0RULE 'upgrade_to_gold' MIGRATE FROM POOL
'silver' WEIGHT(.) TO POOL 'gold' WHERE(.)

[I] Filesystem objects with no applicable rules: 21.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to migrate 0KB: 3 of 3 candidates;
Predicted Data Pool Utilization in KB and %:
Pool_Name              KB_Occupied       KB_Total Percent_Occupied
gold                       3314176        5242880    63.212890625%
silver                       66048        5242880     1.259765625%
system                     1945088        5242880    37.099609375%
[I] 2014-10-13@21:56:37.759 Policy execution. 3 files dispatched.
[I] A total of 3 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
    0 'skipped' files and/or errors.
```

So that the rules are enforced, see Example 5-31 on page 266 which lists the attributes of a file, including to which pool the file belongs.

5.7.4 Example of policies with one external pool only

To feature Spectrum Scale's flexibility, we can define as an external storage pool that runs a custom made script. It works as follows:

1. We define a policy in Spectrum Scale that, based on certain criteria, creates a list of candidate files to be archived.
2. The list is passed to the **`mkiso.sh`** script that creates a DVD ISO image.
3. After the DVD ISO image is created, the files are deleted from the Spectrum Scale file system.

Note: This *archive* policy is only an example as-is and is not meant to be used for production. It only aims to show the flexibility of Spectrum Scale and external pools and not any recommended practice.

This setup has at least the following disadvantages:

- The archived data cannot be recalled. The process works in only one direction.
- Spectrum Scale does not erase the files by itself. An external program has to delete them. After the files are deleted, Spectrum Scale has no knowledge of them. Retrieving them requires a manual process.

- The ISO images are not stored out of the Spectrum Scale file system. A step to move those files to a more permanent storage is missing.

Example 5-45 shows the rule for this example. The internal pool where files reside is called *mypool*. The rule created is called *mypool_to_iso*. The script that is run is located at `/scripts/mkiso.sh`. And the files to be taken into consideration are only those whose name starts with *big*.

Example 5-45 External pool ISO creator policy

```
RULE EXTERNAL LIST 'mkiso' EXEC '/root/scripts/mkiso.sh'
RULE 'mypool_to_iso' LIST 'mkiso' FROM POOL 'system' DIRECTORIES_PLUS WHERE (name) like 'big%
```

Create the `/root/scripts/mkiso.sh` file as shown in Example 5-46. The script functions as follows.

When we apply the Spectrum Scale policy, by using the **`mmapplypolicy`** command, the script is called with two parameters: the command to execute, in our case **`LIST`**, and the path of the temporary file that contains the file's candidates for archiving according to the policy we defined.

For the **`LIST`** parameter, the script takes the files from the Spectrum Scale temporary file list (that contains files to archive) and passes it to the mkisofs DVD ISO image creator. After the DVD ISO image is done, the script erases the files from the list.

Example 5-46 mkiso.sh

```
#!/bin/sh
set -x
case $1 in

     LIST)
        BACKUP_DATE=`date +%F-%H:%M`
        cat $2 | awk '{ print $5 }' >>/tmp/backup_${BACKUP_DATE}.filelist
        mkisofs -o /GPFS/ISO/backup_${BACKUP_DATE}.iso -path-list=/tmp/backup_${BACKUP_DATE}.filelist
                        for i in `cat /tmp/backup_${BACKUP_DATE}.filelist`
                        do
                        rm -fr $i
                        done
        rc=0
        ;;
     TEST)     # Respond with success
        rc=0
        ;;
     *)       # Command not supported by this script
        rc=1
        echo "ERROR: Usage is $0 LIST | TEST"
        ;;
esac
exit $rc
```

Create a file with the contents of Example 5-45, and apply it to the file system as shown in Example 5-47.

Example 5-47 ISO applying the policy

```
# mmapplypolicy GPFS -P iso_policy -L 3
[I] GPFS Current Data Pool Utilization in KB and %
```

```
Pool_Name              KB_Occupied       KB_Total Percent_Occupied
gold                        3314176        5242880    63.212890625%
silver                        66048        5242880     1.259765625%
system                      1945088        5242880    37.099609375%
[I] 4035 of 266880 inodes used: 1.511915%.
[I] Loaded policy rules from iso_policy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-10-13@22:14:29 UTC
Parsed 2 policy rules.
     RULE EXTERNAL LIST 'mkiso' EXEC '/root/scripts/mkiso.sh'
RULE 'mypool_to_iso' LIST 'mkiso' FROM POOL 'system' DIRECTORIES_PLUS WHERE (name) like 'big%'
[I] 2014-10-13@22:14:29.544 Directory entries scanned: 25.
[I] Directories scan: 22 files, 3 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@22:14:29.548 Sorting 25 file list records.
/GPFS/big10113@2 RULE 'mypool_to_iso' LIST 'mkiso' DIRECTORIES_PLUS FROM POOL 'system' WEIGHT(inf)
/GPFS/big102 RULE 'mypool_to_iso' LIST 'mkiso' DIRECTORIES_PLUS FROM POOL 'system' WEIGHT(inf)
[I] Inodes scan: 22 files, 3 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-10-13@22:14:29.585 Policy evaluation. 25 files scanned.
[I] 2014-10-13@22:14:29.593 Sorting 2 candidate file list records.
[I] 2014-10-13@22:14:29.595 Choosing candidate files. 2 records scanned.
[I] Summary of Rule Applicability and File Choices:
 Rule#    Hit_Cnt     KB_Hit     Chosen  KB_Chosen     KB_IllRule
     0          2          0          2          0          0RULE 'mypool_to_iso' LIST 'mkiso'
DIRECTORIES_PLUS FROM POOL 'system' WHERE(.)

[I] Filesystem objects with no applicable rules: 23.

[I] GPFS Policy Decisions and File Choice Totals:
 Chose to list 0KB: 2 of 2 candidates;
Predicted Data Pool Utilization in KB and %:
Pool_Name              KB_Occupied       KB_Total Percent_Occupied
gold                        3314176        5242880    63.212890625%
silver                        66048        5242880     1.259765625%
system                      1945088        5242880    37.099609375%
I: -input-charset not specified, using utf-8 (detected in locale settings).
  4.87% done, estimate finish Tue Oct 14 01:14:29 2014
  9.76% done, estimate finish Tue Oct 14 01:14:29 2014
 14.62% done, estimate finish Tue Oct 14 01:14:29 2014
 19.51% done, estimate finish Tue Oct 14 01:14:29 2014
 24.37% done, estimate finish Tue Oct 14 01:14:29 2014
 29.25% done, estimate finish Tue Oct 14 01:14:29 2014
 34.12% done, estimate finish Tue Oct 14 01:14:29 2014
 39.00% done, estimate finish Tue Oct 14 01:14:31 2014
 43.87% done, estimate finish Tue Oct 14 01:14:31 2014
 48.75% done, estimate finish Tue Oct 14 01:14:31 2014
 53.62% done, estimate finish Tue Oct 14 01:14:30 2014
 58.50% done, estimate finish Tue Oct 14 01:14:30 2014
 63.37% done, estimate finish Tue Oct 14 01:14:30 2014
 68.25% done, estimate finish Tue Oct 14 01:14:30 2014
 73.12% done, estimate finish Tue Oct 14 01:14:30 2014
 78.00% done, estimate finish Tue Oct 14 01:14:31 2014
 82.87% done, estimate finish Tue Oct 14 01:14:31 2014
 87.75% done, estimate finish Tue Oct 14 01:14:31 2014
 92.62% done, estimate finish Tue Oct 14 01:14:31 2014
 97.50% done, estimate finish Tue Oct 14 01:14:31 2014
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
102574 extents written (200 MB)
```

```
[I] 2014-10-13@22:14:31.436 Policy execution. 2 files dispatched.
[I] A total of 2 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
    0 'skipped' files and/or errors.
```

Note: The archive policy is only an example as-is and is not meant to be used for production. It only aims to show the flexibility of Spectrum Scale and external pools and not any recommended practice.

5.8 Access control list

Access control protects directories and files by providing a means of specifying who is granted access. IBM Spectrum Scale access control lists are either traditional access control lists (ACLs) based on the POSIX model, or NFSv4 ACLs. NFSv4 ACLs are very different from traditional ACLs, and provide much more fine control of file and directory access. A Spectrum Scale file system can also be exported by using NFS.

5.8.1 Traditional Spectrum Scale ACL administration

Traditional Spectrum Scale ACLs are based on the POSIX model. Traditional Spectrum Scale ACLs extend the base permissions, or standard file access modes, of read (r), write (w), and execute (x) beyond the three categories of file owner, file group, and other users, to allow the definition of additional users and user groups. In addition, Spectrum Scale introduces a fourth access mode, control (c), which can be used to govern who can manage the ACL itself.

In this way, a traditional ACL can be created that looks like the one in Example 5-48.

Example 5-48 Traditional ACL

```
#owner:jesmith
#group:team_A
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:alpha:r-xc
group:audit:r-x-
group:system:rwx-
```

In the ACL in Example 5-48:

- The first two lines are comments showing the file's owner, jesmith; and group name, team_A.
- The next three lines contain the base permissions for the file. These three entries are the minimum necessary for a Spectrum Scale ACL:
  - The permissions set for the file owner (user), jesmith, *user::rwxc*.
  - The permissions set for the owner's group, team_A, *group::rwx-*.
  - The permissions set for other groups or users outside the owner's group and not belonging to any named entry. The permission `--x-` means that users that are not part of the group or owns the file can only execute it.
- The next line, with an entry type of mask, contains the maximum permissions allowed for any entries other than the owner (the user entry) and those covered by others in the ACL, which are *mask::rwxc*.

- The last three lines contain additional entries for specific users and groups. These permissions are limited by those specified in the mask entry, but you can specify any number of additional entries up to a memory page (approximately 4 K) in size, *user:alpha:r-xc*.

Traditional Spectrum Scale ACLs are fully compatible with the base operating system permission set. Any change to the base permissions, using the **chmod** command, for example, modifies the corresponding Spectrum Scale ACL as well. Similarly, any change to the Spectrum Scale ACL is reflected in the output of commands such as **ls -l**. The control (c) permission is Spectrum Scale-specific. There is no comparable support in the base operating system commands. As a result, the (c) permission is visible only with the Spectrum Scale ACL commands. The control permission is the Spectrum Scale attribute that indicates the user who has authority to change the file or directory attributes and permissions.

Use the **mmputacl** command to set the access ACL of a file or subdirectory, or the default ACL of a directory. For example, to set the ACL for a file named `project2.history`, we can create a file named `project2.acl` that contains the data as shown in Example 5-49.

Example 5-49 project2.acl

```
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:operator:r-xc
group:ftp:r---
group:adm:rwx-
```

Note: If present, the default ACL is used as a base for the access ACL of every object created in that directory. This allows a user to protect all files in a directory without explicitly setting an ACL for each one.

When you are satisfied that the correct permissions are set in the ACL file, apply them to the target file with the **mmputacl** command, as shown in Example 5-50, and **mmgetacl** command, as shown in Example 5-51, to list the ACL of the test directory `project2.history`.

Example 5-50 Setting the ACL

```
# mmputacl -i project2.acl project2.history
```

Example 5-51 shows the listing of the ACL.

Example 5-51 Listing the ACL

```
# mmgetacl project2.history
#owner:root
#group:root
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:operator:r-xc
group:adm:rwx-
group:ftp:r---
```

It is possible to apply the same traditional ACLs from one file or directory to another. To do so:

1. Issue the **mmgetacl** command with the **-o** option to place the information in an output file.
2. Apply the ACLs to the new file or directory by issuing the **mmputacl** command with the **-i** option.

In our example, we want to set the *project3.new* directory the same ACLs as for our project2.history. To do so, we use the **mmgetacl** command and the **mmputacl** command. Copying ACLs from file to file is shown in Example 5-52, Example 5-53, and Example 5-54.

Example 5-52 Get the running ACL to a file

```
# mmgetacl -o all_project.acl project2.history
```

Then, apply it to our new project directory, as shown in Example 5-53.

Example 5-53 Apply it to the new project directory

```
# mmputacl -i all_project.acl project3.new
```

List the ACLs for the new directory, as shown in Example 5-54.

Example 5-54 List the ACLs for the new directory

```
# mmgetacl project3.new/
#owner:root
#group:root
user::rwxc
group::rwx-
other::--x-
mask::rwxc
user:operator:r-xc
group:adm:rwx-
group:ftp:r---
```

Use the **mmdelacl** command to delete the extended entries in a traditional ACL of a file or directory, or the default ACL of a directory. For example, to delete the ACL for the directory `project2.history`, see Example 5-55. Listing the ACLs is shown in Example 5-56.

Example 5-55 Removing ACLs

```
# mmdelacl project2.history
```

Example 5-56 Listing the ACLs

```
# mmgetacl project2.history/
#owner:root
#group:root
user::rwxc
group::rwx-
other::--x-
```

Match the POSIX values as shown by the operating system. See Example 5-57.

Example 5-57 Matching the POSIX values as shown by the operating system

```
# ls -al project2.history
drwxrwx--x 2 root root 4096 Oct 14 16:45 .
drwx------ 5 root root 4096 Oct 14 17:52 ..
```

5.8.2 NFSv4 ACL administration

AIX does not allow a file system to be NFSv4 exported unless it supports NFSv4 ACLs. By contrast, Linux does not allow a file system to be NFSv4 exported unless it supports POSIX ACLs. This is because NFSv4 Linux servers handle NFSv4 ACLs by translating them into POSIX ACLs.

With AIX, the file system must be configured to support NFSv4 ACLs (with the **-k all** or **-k nfs4** option of the **mmcrfs** or **mmchfs** command). The default for the **mmcrfs** command is **-k all**.

With Linux, the file system must be configured to support POSIX ACLs (with the **-k all** or **-k posix** option of the **mmcrfs** or **mmchfs** command).

Note: In general, constructing NFSv4 ACLs is more complicated than traditional ACLs. Users new to NFSv4 ACLs might find it useful to start with a traditional ACL and allow either **mmgetacl** or **mmeditacl** to provide the NFSv4 translation, using the **-k nfs4** flag as a starting point when creating an ACL for a new file.

NFSv4 ACLs are represented in a completely different format than traditional ACLs. For detailed information about NFSv4 and its ACLs, refer to the paper, *NFS Version 4 ACLs*, found at the following site:

https://tools.ietf.org/html/draft-falkner-nfsv4-acls-00

IBM Spectrum Scale has exceptions and limitations to the NFSv4 ACLs that you need to understand. Those exceptions and limitations include:

- Alarm type ACL entries are not supported.
- Audit type ACL entries are not supported.
- Some types of access for which NFSv4 defines controls do not currently exist in Spectrum Scale. For these, ACL entries will be accepted and saved, but since there is no corresponding operation they will have no effect. These include READ_NAMED, WRITE_NAMED, and SYNCHRONIZE.
- AIX requires that READ_ACL and WRITE_ACL always be granted to the object owner. Although this contradicts NFS Version 4 Protocol, it is viewed that this is an area where users would otherwise erroneously leave an ACL that only privileged users could change. Since ACLs are themselves file attributes, READ_ATTR and WRITE_ATTR are similarly granted to the owner. Since it would not make sense to then prevent the owner from accessing the ACL from a non AIX node, Spectrum Scale has implemented this exception everywhere.
- AIX does not support the use of special name values other than owner@, group@, and everyone@. Therefore, these are the only valid special name values for use in Spectrum Scale NFSv4 ACLs as well.
- NFSv4 allows ACL entries that grant users (or groups) permission to change the owner or owning group of the file (for example, with the **chown** command). For security reasons, Spectrum Scale now restricts this so that non-privileged users may only **chown** such a file to themselves (becoming the owner) or to a group that they are a member of.
- Windows operating system does not support NFSv4 ACLs.
- Concurrent Samba, AIX NFS servers, and Spectrum Scale Windows operating system nodes in the cluster are allowed. NFSv4 ACLs can be stored in Spectrum Scale file systems using Samba exports, NFSv4 AIX servers, Spectrum Scale Windows operating

system nodes, **aclput**, and **mmputacl**. However, clients of Linux V4 servers will not be able to see these ACLs, just the permissions from the mode.

Note: In the case of NFSv4 ACLs, there is no concept of a default ACL. Instead, there is a single ACL and the individual ACL entries can be flagged as being inherited (either by files, directories, both, or neither). Consequently, specifying the **-d** flag on the **mmputacl** command for an NFSv4 ACL is an error.

There is no option on the **mmputacl** command to identify the type (traditional or NFSv4) of ACL that is to be assigned to a file. Instead, the ACL is assumed to be in the traditional format unless the first line of the ACL is *#NFSv4 ACL*. In our example, the NFSv4 ACL input file called `nfsv4.acl` contains the data shown in Example 5-58.

Example 5-58 NFSv4 ACL input file

```
#NFSv4 ACL
#owner:root
#group:system
special:owner@:rwxc:allow
(X)READ/LIST (X)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
(X)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED

special:owner@:-----:deny
(-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED
(-)DELETE    (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (X)WRITE_NAMED

user:operator:r-xc:allow
(X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
(X)DELETE    (-)DELETE_CHILD (-)CHOWN (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED

user:operator:-----:deny
(-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED
(-)DELETE    (X)DELETE_CHILD (X)CHOWN (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED
```

Now you can apply the NFSv4 ACLs from file `nfsv4.acl` in Example 5-59 with the **mmputacl** command, and list them with the **mmgetacl** command as shown in Example 5-60.

Example 5-59 Setting NFSv4 ACLs

```
# mmputacl -i nfsv4.acl project4
```

Example 5-60 Listing the ACLs

```
# mmgetacl project4
#NFSv4 ACL
#owner:root
#group:root
special:owner@:rwxc:allow
 (X)READ/LIST (X)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
 (X)DELETE    (-)DELETE_CHILD (-)CHOWN        (X)EXEC/SEARCH (X)WRITE_ACL (X)WRITE_ATTR (-)WRITE_NAMED

special:owner@:-----:deny
 (-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED
 (-)DELETE    (X)DELETE_CHILD (X)CHOWN        (-)EXEC/SEARCH (-)WRITE_ACL (-)WRITE_ATTR (X)WRITE_NAMED

user:operator:r-xc:allow
 (X)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (X)SYNCHRONIZE (X)READ_ACL  (-)READ_ATTR  (-)READ_NAMED
 (X)DELETE    (-)DELETE_CHILD (-)CHOWN        (X)EXEC/SEARCH (X)WRITE_ACL (-)WRITE_ATTR (-)WRITE_NAMED
```

```
user:operator:----:deny
 (-)READ/LIST (-)WRITE/CREATE (-)APPEND/MKDIR (-)SYNCHRONIZE (-)READ_ACL  (-)READ_ATTR  (X)READ_NAMED

 (-)DELETE    (X)DELETE_CHILD (X)CHOWN        (-)EXEC/SEARCH (-)WRITE_ACL (X)WRITE_ATTR (X)WRITE_NAMED
```

Use the **mmdelacl** command to delete NFSv4 ACLs. When the ACL has been deleted, permissions revert to the mode bits. If the **mmgetacl** command is then used to display the ACL, it appears as a traditional Spectrum Scale ACL.

To delete the NFSv4 ACL in our example, see Example 5-61.

Example 5-61 Removing NFSv4 ACLs

```
# mmdelacl project4
```

To list the ACLs, see Example 5-62.

Example 5-62 Listing the ACLs

```
# mmgetacl project4
#owner:root
#group:root
user::rwxc
group::----
other::----
```

5.9 The mmfind policy sample

From IBM Spectrum Scale 4.1.0.5 onwards, a new sample has been added to `/usr/lpp/mmfs/samples/ilm`, it is named *mmfind*. It is not part of the official product and lacks any warranty or support from IBM, and it might suffer changes on future updates of the samples directory. The version presented on 4.1.0.5 introduces an alternative to POSIX command find using native Spectrum Scale calls and other tools. It is a find program that is wrapped around the Spectrum Scale **mmapplypolicy** command. You can use it to quickly scan a directory structure and run a script against each file you find. The **mmapplypolicy** command only finds files that have been synced to disk. In large file systems with many files and directories, the **mmfind** tool introduces a speedy alternative to the POSIX **find** command.

Note: All the policies and utilities included inside `/usr/lpp/mfs/samples` are not part of the official IBM Spectrum Scale deployment and are given as-is. Double check what the files do before using it in your systems. No support can be obtained from IBM for any utilities at the samples directory.

It is a rather powerful tool and comes with online help. In Example 5-63 on page 286, a simple usage of the **mmfind** utility is given that lists files that are larger than 150 MBytes.

Note: On the **mmfind** utility released on 4.1.0.5, it is not yet possible to mix more than one search criteria. It has to be either name, owner, size, access time, modification time, and so on. If multiple criteria are selected, the files that hit any of the criteria are then selected. See it as a logical OR not an AND nor XOR. It has to run with *root* privileges.

Example 5-63 mmfind example

```
# mmfind -policyPath /GPFS -exec 'ls -alh FILENAME >> /tmp/mmfind.out' -size +150M -onlyNormalFiles
Sun Nov 16 21:41:04.722679: tid 0: Script mmfind is starting up on node nsdserver01 in pwd </root> with
arguments: -policyPath /GPFS -exec ls -alh FILENAME >> /tmp/mmfind.out -size +150M -onlyNormalFiles
Sun Nov 16 21:41:04.723025: tid 0: CMD: /usr/lpp/mmfs/bin/mmlscluster
Sun Nov 16 21:41:05.446097: tid 0: CMD: /usr/lpp/mmfs/bin/mmgetstate -a -Y
Sun Nov 16 21:41:11.654451: tid 0: CMD: /usr/lpp/mmfs/bin/mmlscluster
Sun Nov 16 21:41:12.649976: tid 0: CMD: /usr/lpp/mmfs/bin/mmlsfs all -T 2>&1
Sun Nov 16 21:41:13.580254: tid 0: I have the following parameters to work with: fs <GPFS> path </GPFS>
nodes <nsdserver01> nofiles <0> noDirs <1> noSymlinks <1> noCloneParents <1> noImmutable <1> noAppendOnly
<1> noEncrypted <0> noUnencrypted <0> name <> iname <> path <> ipath <> xattr <> size <+157286400> inode <>
uid <> gid <> link <> initialPolicyPathDepth <1>
Sun Nov 16 21:41:13.580465: tid 0: Making sure all nodes are in 'active' state
Sun Nov 16 21:41:13.580611: tid 0: CMD: /usr/lpp/mmfs/bin/mmgetstate -N nsdserver01 -Y
Sun Nov 16 21:41:17.102287: tid 0: Making sure fs GPFS is mounted on all nodes <nsdserver01>
Sun Nov 16 21:41:17.102552: tid 0: CMD: /usr/lpp/mmfs/bin/mmmount GPFS -N nsdserver01
Sun Nov 16 21:41:18 EET 2014: mmmount: Mounting file systems ...
Sun Nov 16 21:41:20.144132: tid 0: CMD: /usr/lpp/mmfs/bin/mmlsmount GPFS -L -Y 2>&1
Sun Nov 16 21:41:20.792617: tid 0: Generating helper script for the EXEC field
Sun Nov 16 21:41:20.792965: tid 0: CMD: chmod +x
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript
Sun Nov 16 21:41:20.794536: tid 0: EXEC script is
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript
Sun Nov 16 21:41:20.794624: tid 0: Copying EXEC script to each node in the cluster:
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript
Sun Nov 16 21:41:20.806182: tid 1: CMD: /usr/bin/scp
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript
nsdserver01:/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript
Sun Nov 16 21:41:20.993400: tid 1: CMD: /usr/lpp/mmfs/bin/mmdsh -L nsdserver01 'chmod +x
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript'
Sun Nov 16 21:41:21.259580: tid 0: Generating policy file
Sun Nov 16 21:41:21.259983: tid 0: Policy file is /tmp/mmfind_time1416166864_pid9166_polFile
Sun Nov 16 21:41:21.260038: tid 0: Invoking mmapplypolicy
Sun Nov 16 21:41:21.260088: tid 0: runcmdLiveOutput:  /usr/lpp/mmfs/bin/mmapplypolicy /GPFS -P
/tmp/mmfind_time1416166864_pid9166_polFile -N nsdserver01
Sun Nov 16 21:41:21.261875: tid 0: runcmdLiveOutput: subcmd has pid 10004
Sun Nov 16 21:41:21.800644: tid 0: runcmdLiveOutput:  [I] GPFS Current Data Pool Utilization in KB and %
Sun Nov 16 21:41:21.800776: tid 0: runcmdLiveOutput:  Pool_Name              KB_Occupied       KB_Total
Percent_Occupied
Sun Nov 16 21:41:21.800817: tid 0: runcmdLiveOutput:  gold                       3314176        5242880
63.212890625%
Sun Nov 16 21:41:21.800856: tid 0: runcmdLiveOutput:  silver                       66048        5242880
1.259765625%
Sun Nov 16 21:41:21.800892: tid 0: runcmdLiveOutput:  system                     2150656        5242880
41.020507812%
Sun Nov 16 21:41:21.800927: tid 0: runcmdLiveOutput:  [I] 4042 of 266880 inodes used: 1.514538%.
Sun Nov 16 21:41:21.801467: tid 0: runcmdLiveOutput:  [I] Loaded policy rules from
/tmp/mmfind_time1416166864_pid9166_polFile.
Sun Nov 16 21:41:21.801520: tid 0: runcmdLiveOutput:  Evaluating policy rules with CURRENT_TIMESTAMP =
2014-11-16@19:41:21 UTC
Sun Nov 16 21:41:21.801556: tid 0: runcmdLiveOutput:  Parsed 3 policy rules.
Sun Nov 16 21:41:21.801588: tid 0: runcmdLiveOutput:
Sun Nov 16 21:41:21.801621: tid 0: runcmdLiveOutput:  /* exclude anything we want to specifically exclude */
Sun Nov 16 21:41:21.801654: tid 0: runcmdLiveOutput:  RULE 'exclude_stuff' LIST 'all'
Sun Nov 16 21:41:21.801684: tid 0: runcmdLiveOutput:  DIRECTORIES_PLUS
Sun Nov 16 21:41:21.801719: tid 0: runcmdLiveOutput:  EXCLUDE WHERE FALSE   OR ( MODE LIKE 'd%' )  OR ( MODE
LIKE 'l%' )  OR ( MODE LIKE 'c%' )  OR ( MODE LIKE 'b%' )  OR ( MODE LIKE 'p%' )  OR ( MODE LIKE 's%' )  OR
( MODE LIKE '?%' )  OR ( (((xattr_integer('gpfs.CLONE',13,1,'B')+256)/128)%2) IS NOT NULL AND
(((xattr_integer('gpfs.CLONE',13,1,'B')+256)/128)%2) == 1 )  OR ( MISC_ATTRIBUTES LIKE '%a%' )  OR (
MISC_ATTRIBUTES LIKE '%X%' )  OR ( FILE_SIZE <= (157286400) )
```

```
Sun Nov 16 21:41:21.801791: tid 0: runcmdLiveOutput:
Sun Nov 16 21:41:21.801829: tid 0: runcmdLiveOutput:  /* list everything else */
Sun Nov 16 21:41:21.801864: tid 0: runcmdLiveOutput:  RULE 'list_the_rest' LIST 'all'
Sun Nov 16 21:41:21.801899: tid 0: runcmdLiveOutput:  DIRECTORIES_PLUS
Sun Nov 16 21:41:21.801928: tid 0: runcmdLiveOutput:
Sun Nov 16 21:41:21.801956: tid 0: runcmdLiveOutput:  /* run all non-excluded files into the auto-generated
external script */
Sun Nov 16 21:41:21.801991: tid 0: runcmdLiveOutput:  RULE 'ext' EXTERNAL LIST 'all'
Sun Nov 16 21:41:21.802025: tid 0: runcmdLiveOutput:    EXEC
'/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript'
Sun Nov 16 21:41:21.802058: tid 0: runcmdLiveOutput:    ESCAPE '%' /* URI encode all of the characters so we
can use &uriDecode to decode them cleanly */
Sun Nov 16 21:41:21.820297: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.819 Directory entries scanned:
0.
Sun Nov 16 21:41:21.827025: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.826 Directory entries scanned:
32.
Sun Nov 16 21:41:21.827159: tid 0: runcmdLiveOutput:  [I] Directories scan: 26 files, 6 directories, 0 other
objects, 0 'skipped' files and/or errors.
Sun Nov 16 21:41:21.827214: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.826 Sorting 32 file list
records.
Sun Nov 16 21:41:21.830756: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.829 Sorting 32 file list
records.
Sun Nov 16 21:41:21.830933: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.829 Policy evaluation. 0 files
scanned.
Sun Nov 16 21:41:21.893537: tid 0: runcmdLiveOutput:  [I] Inodes scan: 26 files, 6 directories, 0 other
objects, 0 'skipped' files and/or errors.
Sun Nov 16 21:41:21.893816: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.893 Policy evaluation. 32
files scanned.
Sun Nov 16 21:41:21.894158: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.893 Sorting 5 candidate file
list records.
Sun Nov 16 21:41:21.898324: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.897 Sorting 5 candidate file
list records.
Sun Nov 16 21:41:21.898464: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.897 Choosing candidate files.
0 records scanned.
Sun Nov 16 21:41:21.898500: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.898 Choosing candidate files.
5 records scanned.
Sun Nov 16 21:41:21.899032: tid 0: runcmdLiveOutput:  [I] Summary of Rule Applicability and File Choices:
Sun Nov 16 21:41:21.899230: tid 0: runcmdLiveOutput:   Rule#    Hit_Cnt     KB_Hit     Chosen  KB_Chosen
KB_IllRule
Sun Nov 16 21:41:21.899271: tid 0: runcmdLiveOutput:       0         27     102656          0          0
0   RULE 'exclude_stuff' LIST 'all' EXCLUDE DIRECTORIES_PLUS WHERE(.)
Sun Nov 16 21:41:21.899296: tid 0: runcmdLiveOutput:       1          5    4399456          5    4399456
0   RULE 'list_the_rest' LIST 'all' DIRECTORIES_PLUS
Sun Nov 16 21:41:21.899622: tid 0: runcmdLiveOutput:
Sun Nov 16 21:41:21.899677: tid 0: runcmdLiveOutput:  [I] Filesystem objects with no applicable rules: 0.
Sun Nov 16 21:41:21.899703: tid 0: runcmdLiveOutput:
Sun Nov 16 21:41:21.899722: tid 0: runcmdLiveOutput:  [I] GPFS Policy Decisions and File Choice Totals:
Sun Nov 16 21:41:21.899741: tid 0: runcmdLiveOutput:   Chose to list 4399456KB: 5 of 5 candidates;
Sun Nov 16 21:41:21.899786: tid 0: runcmdLiveOutput:  Predicted Data Pool Utilization in KB and %:
Sun Nov 16 21:41:21.899825: tid 0: runcmdLiveOutput:  Pool_Name              KB_Occupied       KB_Total
Percent_Occupied
Sun Nov 16 21:41:21.899868: tid 0: runcmdLiveOutput:  gold                       3314176        5242880
63.212890625%
Sun Nov 16 21:41:21.899902: tid 0: runcmdLiveOutput:  silver                       66048        5242880
1.259765625%
Sun Nov 16 21:41:21.899935: tid 0: runcmdLiveOutput:  system                     2150656        5242880
41.020507812%
Sun Nov 16 21:41:21.899967: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.899 Policy execution. 0 files
dispatched.
```

```
Sun Nov 16 21:41:21.954343: tid 0: runcmdLiveOutput:  [I] 2014-11-16@19:41:21.954 Policy execution. 5 files
dispatched.
Sun Nov 16 21:41:21.954454: tid 0: runcmdLiveOutput:  [I] A total of 5 files have been migrated, deleted or
processed by an EXTERNAL EXEC/script;
Sun Nov 16 21:41:21.954483: tid 0: runcmdLiveOutput:  0 'skipped' files and/or errors.
Sun Nov 16 21:41:21.959086: tid 0: runcmdLiveOutput: Command done, no more output is coming
Sun Nov 16 21:41:21.959416: tid 0: Cleaning up -- deleting tmp files
Sun Nov 16 21:41:21.959539: tid 0: CMD: /usr/lpp/mmfs/bin/mmdsh -N nsdserver01 'rm
/tmp/mmfind_start1416166864_hostnsdserver01_pid9166_execScript > /dev/null 2>&1; rm
/tmp/mmfind_per_node_file_list_1416166864 > /dev/null 2>&1' > /dev/null 2>&1
Sun Nov 16 21:41:22.336303: tid 0: All done, have a nice day

# cat /tmp/mmfind.out
-rw-r--r-- 1 root root 1.0G Oct 14 00:25 /GPFS/1G.zero
-rw-r--r-- 1 root root 1.0G Oct 14 00:26 /GPFS/1G.zero.1
-rw-r--r-- 1 root root 1.0G Oct 14 00:27 /GPFS/1G.zero.2
-rw-r--r-- 1 root root 1.0G Oct 14 00:28 /GPFS/1G.zero.3
-rw-r--r-- 1 root root 201M Oct 14 01:19 /GPFS/ISO/backup_2014-10-14-01:19.iso
```

5.10 Differences with IBM EasyTier

Easy Tier is a no-charge feature of the IBM DS8000 and IBM Spectrum Virtualize systems. However, as with any other acquired licensed function, the Easy Tier licensed functions must first be ordered from IBM.

Automatic mode

Easy Tier Automatic Mode automatically (see Figure 5-7) optimizes storage performance and storage economics management across drive tiers through data placement on a subvolume level in multitier or hybrid extent pools. Multitier or hybrid extent pools are storage pools that contain a mix of different drive tiers. Easy Tier can nondisruptively relocate data at extent level across different drive tiers or even within the same drive tier to optimize performance and resource usage. The auto-rebalance capability automatically rebalances the workload, even across the physical resources within a drive tier, which reduces the occurrence of hotspots.

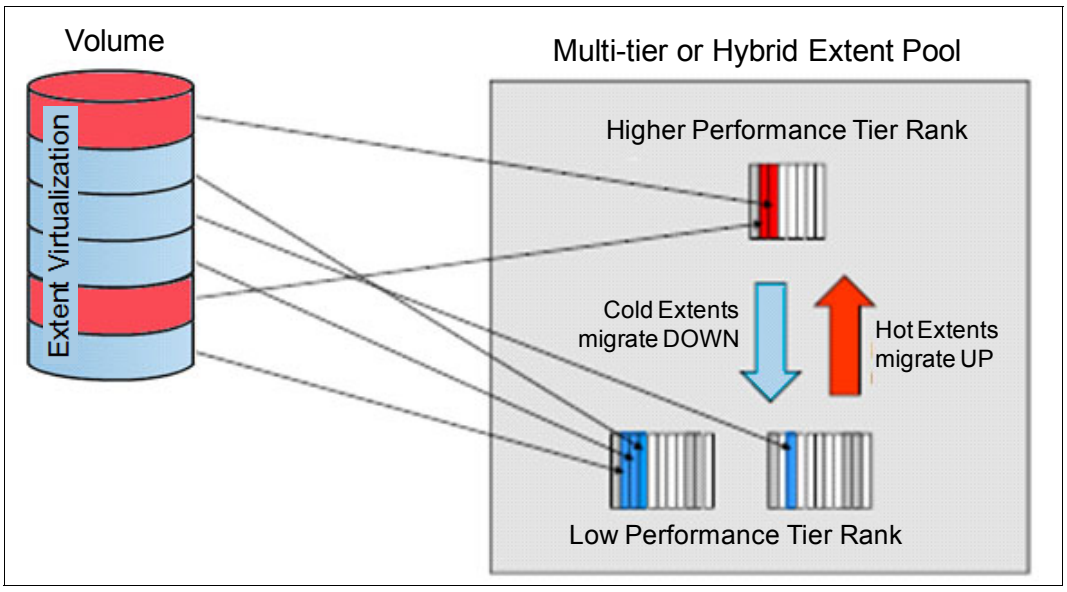


Figure 5-7 Easy Tier Automatic Mode

Manual Mode

Easy Tier also has a Manual Mode, which covers user-requested functions that allow you to manually merge extent pools, depopulate ranks, and relocate volumes. This mode applies to single-tier (homogeneous) and multitier (hybrid) extent pools. In Manual Mode, volumes can be relocated between extent pools to balance the performance across the extent pools or to move a volume to a more appropriate extent pool, such as a flash rank or a hybrid extent pool.

Note: Easy Tier Automatic Mode and Manual Mode are not exclusive, which means that you can use Manual Mode capabilities even if Automatic Mode is active.

For more information about IBM Easy Tier, see the IBM Redpaper publication *IBM DS8000 Easy Tier*, REDP-4667, which can be found at the following website:

`http://www.redbooks.ibm.com/abstracts/redp4667.html?Open`

IBM Easy Tier does its move between defined tiers at extend level. It does not know about the file system or its attributes. IBM Spectrum Scale ILM can move data knowing exactly what data is used for, by which users or groups, and if the data is only being listed or being used. IBM Easy Tier is an excellent companion and complement of IBM Spectrum Scale ILM.

6

Active File Management

This chapter provides common use cases for IBM Spectrum Scale Active File Management (AFM) including connecting clusters together, migrating data from existing NFS storage, and creating a global namespace across multiple Spectrum Scale clusters.

AFM was introduced in GPFS 3.5 originally as a method of remotely caching data. Since then, it has proven useful for many different types of asynchronous data movement.

The following topics are included:

- Active file management fundamentals
- AFM single-writer
- AFM independent-writer
- Using AFM to migrate the content of an existing file system
- Customizing and tuning AFM filesets
- Building a global name space
- Enhancements in IBM Spectrum Scale 4.1 TL 1

© Copyright IBM Corp. 2015. All rights reserved.

6.1 Active file management fundamentals

GPFS 3.5 introduced a new feature that enables sharing of data across clusters, even if the networks are unreliable or have high latencies. This feature is called *active file management* (AFM). AFM allows you to create associations between IBM Spectrum Scale clusters and automate the location and flow of file data. With AFM, you can implement a single name space view across sites around the world making your global name space truly global.

AFM is a scalable, high-performance, file system caching feature that is part of Spectrum Scale. AFM is capable of masking wide area network (WAN) latencies and outages by allowing you to automate the control of data location and data copies, independent of the total number of objects in the namespace.

Namespace maintenance with AFM occurs asynchronously, which allows applications to continue operating while not being constrained by network bandwidth. AFM can be used to build a common namespace across different locations and platforms and it can be used for duplicating data for disaster recovery purposes without suffering from WAN latencies. AFM is an active manager constantly keeping track of changes in the Cache and at the HOME. Changes are managed per fileset resulting in a modular, scalable architect capable of supporting billions of files and many petabytes of data.

This chapter is a quick start guide introducing the AFM features of Spectrum Scale so you can discover quickly how to leverage these features in your environment. This includes some common approaches to building a global name space spanning locations, and how to use AFM to keep sites synchronized. AFM capabilities are powerful, hence this chapter is not intended to provide complete documentation of all the features.

With the introduction of AFM, some new terminology has been added to IBM Spectrum Scale. The granularity of an AFM container is a fileset. Each fileset can have a distinct set of AFM attributes. A Spectrum Scale cluster that contains AFM-enabled filesets is called a *Cache cluster*. A Cache fileset has a relationship with a data source called the *HOME*. The HOME can be another Spectrum Scale file system accessible by NFS or the Network Shared Disk (NSD) protocol or any NFS accessible storage. If HOME is not a Spectrum Scale file system, there are some limitations on functionality (for example no parallel data transfers). Figure 6-1 on page 293 shows the relationships between AFM components. AFM fileset definition is flexible, which means a Cache cluster for one set of data can be a HOME for another set of data.

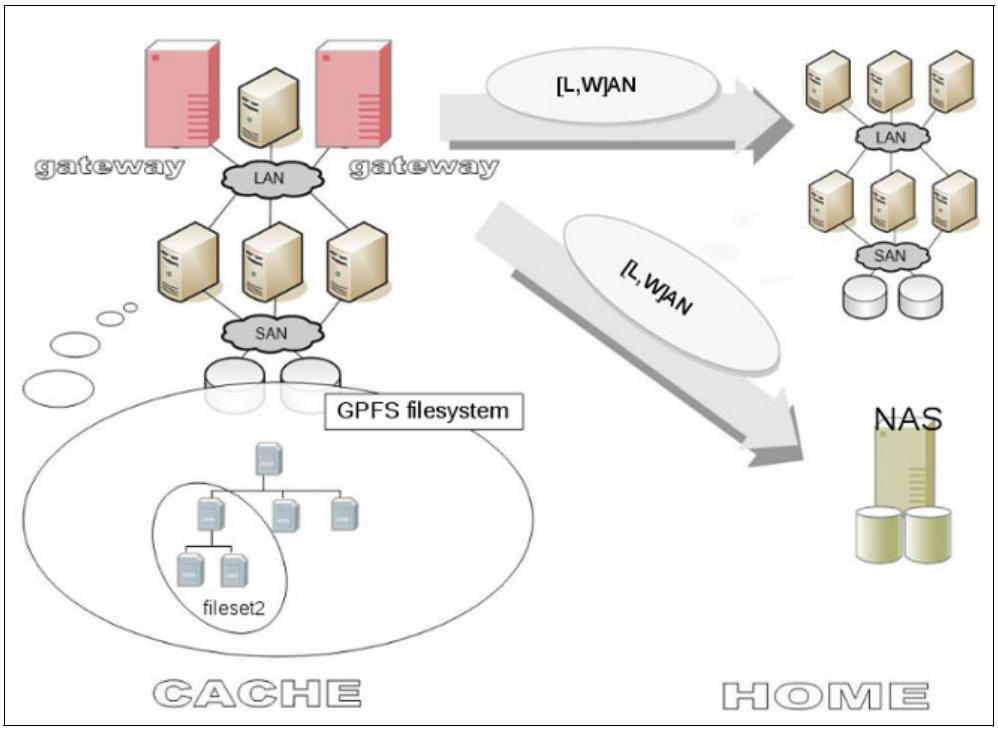


Figure 6-1 IBM Spectrum Scale AFM overview and terminology

Data movement to and from the Cache is done through one or more gateway nodes. At the time this publication was written, only the Linux operating system is supported for a gateway node. There is a gateway node designated at the “owner” of each Cache fileset. That node is responsible for maintaining the list of changes that need to be replicated to the HOME. The gateway function is highly available and scale-out. If a gateway node fails, the ownership of a Cache fileset fails over to another gateway node. The primary gateway node can get help from other gateway nodes for parallel data movement to and from HOME supporting very high performance data transfers.

Any Spectrum Scale cluster can be a HOME cluster, a Cache cluster, or both. In typical implementations, HOME is created in one Spectrum Scale cluster and a Cache in another, but this placement is not required. In fact, a cluster can be a HOME for one fileset and a Cache for another fileset. Multiple AFM-enabled filesets can be defined in one Cache cluster, each caching data from a different HOME cluster. This implementation provides great flexibility in how you can leverage the caching behavior.

6.1.1 AFM mode

AFM provides different Cache behaviors depending on how you would like data to flow. These are called *AFM Cache modes*. Table 6-1 on page 294 shows the AFM caching modes that are available and lists Cache modes for transferring data between HOME and Cache. Filesets in local-update or read-only mode pull data from HOME, while single-writer or independent-writer pull data and push changed data automatically to HOME.

Generally, we decide between two AFM modes: Filesets in local-update (LU) or read-only (RO) mode working in a kind of pull mechanism that means only data that is accessed from the Cache side is read from HOME, and is kept a certain time interval as valid in the Cache. There are two other modes working in a kind of push mode, which means that once the data is written to the Cache, it is transferred by AFM to HOME in the background automatically.

Table 6-1 AFM valid fileset modes

Fileset mode	Description
Read-only (RO)	► Data in the Cache is read-only. Creating or modifying files in the Cache fileset is not allowed.
Local-update (LU)	► Cached data (data from HOME) is read-only. The difference from read-only mode is that you can create and modify files in the Cache fileset. Files that are created or modified in the Cache are considered local updates. Local updates are never pushed back to HOME. ► Once a file is modified in the Cache, the file is no longer compared to the version at HOME to verify that it is up to date. ► Appending to, truncating, or writing to an uncached file fetches the entire file to Cache before making the change locally. **Note:** Any small change done in the LU fileset directory could cause the fileset to be marked as local and lose context with the HOME. For example, running **chmod** or **chown** of the LU fileset root directory causes the entire fileset to be out of sync with the HOME.
Single-writer (SW)	► One Cache fileset does all the writing. The assumption is that all of the other Cache filesets associated with the same HOME are configured as read-only or local-update. This mode avoids possible write conflicts. There is no explicit enforcement of a single-writer mode in Spectrum Scale. Rather, the administrator needs to guarantee that there are no other writable Caches associated with this HOME and that no writes are done at HOME. ► On single-writer filesets, a write at HOME is not allowed. However, if there is a write/corruption at HOME, inadvertently or due to some error situation, it results in a conflicted scenario for the single-writer fileset. It might be repaired automatically but it can force an administrator to re-create the AFM relationship to a new HOME and retransferring all the data from Cache again in background.
Independent-writer (IW)	► Allows multiple Caches to write to different files. There is no locking between clusters. Each Cache makes its updates to HOME independently. Independent-writer mode supports transparent synchronization with HOME. That is, changes in IW Cache are synchronized with HOME and vice versa. In case data is updated at HOME, all connected IW Caches fetch those changes on demand based on the revalidation intervals set. ► Multiple active Cache filesets point to one HOME file system or fileset and can modify the data at the same time. This mode should only be used by applications running in Caches that do not require synchronization to properly update the data on the HOME cluster. ► The multiple Cache sites do revalidation periodically and pull the new data from HOME. There is no locking or ordering between updates; the updates are propagated to the HOME in an asynchronous manner and may be delayed due to network disconnections. Therefore, conflicting updates from multiple Cache sites can cause the data at the HOME site to be undetermined.

6.1.2 Minimum requirements and basic operating system tuning

Since AFM is available with GPFS 3.5, there are several enhancements depending on the minimum Spectrum Scale level. Refer to Table 6-2 on page 296.

In addition, some of the operating system settings need adjusting to take advantage of stable working AFM capabilities. According to your network-specific round-trip times, a summary is provided as a good starting point for tuning the operating system (OS). In addition, point out that the minimum number of free memory pages is set accordingly. The following examples give you a good starting point on how to set up your environment. Additionally, you find a suggestion for adjusting the appropriate settings under a Linux operating system. In case the NFS protocol is used to perform the data transfer for AFM, remember to configure your local firewall settings accordingly.

One thing to note is that the gateway node might need larger memory due to memory demand for server performance and in recovery times. Therefore, it is recommended to

provide sufficient memory according to your expected workload. For minimum acceptable operations, it is required to provide at least 2 GB for the gateway node.

The size of your TCP network TCP send/receive buffer should be set according to your latency. The relationship is easy to calculate with the following note.

For best performance on high latency networks, tune the TCP send/receive buffers according to the latency (round-trip time) on your network. You can calculate the buffer size that is needed to get full throughput by dividing the throughput goal by the round-trip time in seconds.

Note: required TCP buffer size [MB] network bandwidth [MBps] * latency [s].

Example 6-1 shows a sample configuration for a round-trip time of 100 ms (0.1 seconds) with a throughput goal of 10 Gbps (1200 MBps), 1200 MBps * 0.1sec = 120 MB buffer size.

Example 6-1 OS/Linux tuning for AFM

```
sysctl -w net.ipv4.tcp_rmem= *** according to your network latencies
sysctl -w net.ipv4.tcp_wmem= *** according to your network latencies
sysctl -w net.ipv4.tcp_mem= *** accord...
sysctl -w net.core.rmem_max=12194304*
sysctl -w net.core.wmem_max=12194304*
sysctl -w net.core.rmem_default=12194304*
sysctl -w net.core.wmem_default=12194304*
sysctl -w net.core.optmem_max=12194304*
# The following adjustments should set, regardless of a given LAN or WAN latency
sysctl -w net.core.netdev_max_backlog=50000
sysctl -w net.ipv4.tcp_no_metrics_save=1
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_sack=1
sysctl -w vm.min_free_kbytes= [minimum 5-6 %]RealMem
```

Note: Be careful about the min. free setting in your operating system. Additionally, keep in mind that in virtualized environments, the amount of memory might be changed dynamically. Therefore, it is advised to add enough margin to your settings, that at least 5% of the memory remains in the free list of your system.

A good tuning summary is documented in the Advanced Administration Guide at this web page:

http://www-01.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html

and also at the following website:

http://www-01.ibm.com/support/knowledgecenter/SSFKCN/gpfs4104/com.ibm.cluster.gpfs.v4r104.gpfs200.doc/bl1adv_afmtuning.htm

To get started, the important changes are shown in Example 6-1. Table 6-2 on page 296 shows an overview of AFM enhancements and features with certain levels of GPFS/IBM Spectrum Scale.

Table 6-2 AFM features and GPFS/IBM Spectrum Scale release levels

GPFS/IBM Spectrum Scale versions	Related AFM features and enhancements
GPFS 3.5.0.0	Introduces AFM in GPFS: ► Includes single-writer, read-only, and local updates as Cache options. ► Limited supported on x86 Linux only.
GPFS 3.5.0.11 (TL3)	New Cache mode independent-writer. Partial file caching, which might be useful if you have workloads that need to read only a small range of data large file from HOME. Support for AIX: This means that an AIX system can participate in a cluster supporting Cache filesets. The gateway node still runs in an x86_64 based Linux node.
GPFS 4.1.0.0	NSD Protocol Support: In addition to the NFS protocol, AFM now supports the use of the NSD protocol for data transfers. This provides improved integration of Spectrum Scale features and attributes. Various features and optimization for AFM usability, including prefetch enhancements.
GPFS 4.1.0.4 (TL1)	AFM environments now support parallel data transfers between the gateway nodes and HOME. AFM can now spread the workload over all available gateway nodes. A new AFM hash version better balances parallel data transfers. AFM now supports the migration of data from earlier NFS storage devices to Spectrum Scale (within an AFM fileset).

At the time this publication was written, some limitations are known as follows: Having Windows operating system nodes in your cluster may lead to unexpected behavior. In addition, there is an APAR in progress that LU and RO filesets are not aware of default placement policies from the underlaying GPFS file system. If you intend to use AFM filesets within a GPFS file system, which has separate storage pools for metadata and data, check that you have the appropriate fix installed.

Note: Access to an AFM-enabled fileset from a Windows operating system node is not yet supported.

6.1.3 Using NFS for data movement

Keep in mind that the NFS server settings on HOME might be slightly different according to your operating system. When possible use a highly available NFS solution. In GPFS 4.1 and earlier with Linux, you can use the clustered NFS (cNFS) feature. For the Linux HOME system, some tuning suggestions are included in Example 6-2 on page 297. If your HOME is in an AIX system, the default values for the NFS server work fine. The system settings concerning the memory and file Cache policies defaults to a low limit. If there are concerns about your application workload and resources of your system, it might be useful to adjust the *minperm*, *maxclient*, and *maxperm*.

Example 6-2 Tuning the NFS server in Linux

```
/proc/fs/nfsd/max_block_size      # Set this to 1MB for improved performance.
/proc/fs/nfsd/threads             # Set to a minimum value of 32, but possibly
                                    greater than 64 depending on the throughput
```

To improve transfer performance, set the Spectrum Scale **nfsPrefetchStrategy** parameter as shown in Example 6-3. If your HOME is in an earlier NFS server, you can ignore this step.

Example 6-3 Tuning Spectrum Scale in an NFS server

```
mmchconfig nsfPrefetchStrategy=5
```

If you are creating a location for a Spectrum Scale HOME, it may be a good idea to place the HOME data in an independent fileset. If HOME data exists, define an NFS export at the appropriate base directory.

According to the Cache mode, which is configured later on the Cache side, HOME might not be changed anymore from the moment the Cache is activated for the first time. Refer to Example 6-4.

Example 6-4 Preparing HOME and optionally create a fileset

```
# having a gpfs filesystem ready for use
node4# mount | grep gpfs
/dev/fshome on /gpfs/fshome type gpfs (rw,mtime,dev=fshome)

node4# create a fileset, this is optional for the HOME side
node4# mmcrfileset fshome fset001 --inode-space=new
Fileset fset001 created with id 1 root inode 65539.
node4:/usr/lpp/mmfs/src # mmlsfileset fshome
Filesets in file system 'fshome':
Name                    Status    Path
root                       Linked    /gpfs/fshome
fset001                 Unlinked  --

# now link the fileset
nnode4:/usr/lpp/mmfs/src # mmlinkfileset fshome fset001 -J /gpfs/fshome/fset001/
Fileset fset001 linked at /gpfs/fshome/fset001
node4:#
```

Use the **mmafmconfig** command to enable support for ACLs and extended attributes (EAs) to be transferred between HOME and Cache. Run the command as shown in Example 6-5 on page 298 for the root directory of the HOME exported path. The command syntax changed between GPFS 3.5 and GPFS 4.1. If you are using GPFS 3.5 or earlier, use the **mmafmhomeconfig** instead of **mmafmconfig**.

After you have successfully initiated this command, some special files appear in your directory. These directories must not be removed. Since ACLs are copied between HOME and Cache, it is recommended that the user IDs and group IDs on the Cache and HOME are identical, otherwise ACLs might not work as expected.

If you use an earlier NFS source transferring EA and non-NFS (POSIX), ACLs are not supported.

Example 6-5 Enable HOME for ACL/EA transport

```
node4:/gpfs/fshome/fset001 # mmlsfileset fshome   # just to show our entry level in the path
Filesets in file system 'fshome':
Name                    Status    Path
root                       Linked    /gpfs/fshome
fset001                 Linked    /gpfs/fshome/fset001

# now do the enablement
node4:/gpfs/fshome/fset001 # mmafmconfig enable /gpfs/fshome/fset001

#verify the new created directories,
node4:/gpfs/fshome/fset001 # ls -l
total 1
drwxr-xr-x 3 root root  4096 Oct  7 23:12 .afm
dr-xr-xr-x 2 root root 32768 Jan  1  1970 .snapshots
node4:/gpfs/fshome/fset001 #
```

If you use a non-Spectrum Scale source or you do not enable AFM HOME for ACLs, you will see daemon warnings (Example 6-6) in your mmfslog on the Cache side when you start working with your Cache filesets.

Example 6-6 mmfslog messages, when HOME is not enabled for AFM

```
Fri Oct 10 23:52:25.747 2014: [I] AFM: Cannot find control file for file system fs1 fileset single_writer in the exported file system at home. ACLs and extended attributes will not be synchronized. Sparse files will have zeros written for holes.
```

Next, check that the NFS server is running and that the export configuration is correct. NFS is supported in version 3 and 4. Remember to enter a valid fsid if you are exporting a Spectrum Scale file system. Example 6-7 is an example of a valid `/etc/exports` file under Linux. The node list that you export the directories to must contain all the gateway nodes of the Cache clusters. Now you are ready to create Cache filesets.

Example 6-7 Sample NFS export configuration

```
node4:/gpfs # cat /etc/exports | tail -1
/gpfs/fshome/fset001 node2(rw,no_root_squash,no_subtree_check,fsid=101)
node4:/gpfs # exportfs -a
node4:/gpfs # exportfs
/gpfs/fshome/fset001
                node2.site
node4:/gpfs #
```

Note: Ensure that the NFS server is working properly and the exported directory is accessible from all gateway nodes in all Cache clusters.

6.2 AFM single-writer

In this section, the steps to create an AFM single-writer mode are described.

6.2.1 Creating a single-writer Cache fileset

Single-writer (SW) mode is useful when the Cache is the sole source for a particular set of data. For example, if you have a gene sequencing system generating data in a remote location and the data is sent to a central site for analysis, keeping the data for this device in a single-writer AFM fileset makes sense. Single-writer and independent-writer are very similar. The difference between these modes is that changes are tracked by AFM. A single-writer fileset never checks for changes on HOME and an independent-writer fileset does, so network traffic to check for changes is less with single-writer.

After setting up HOME, refer to 6.1.3, “Using NFS for data movement” on page 296. You can now create a Cache fileset as shown in Figure 6-2.

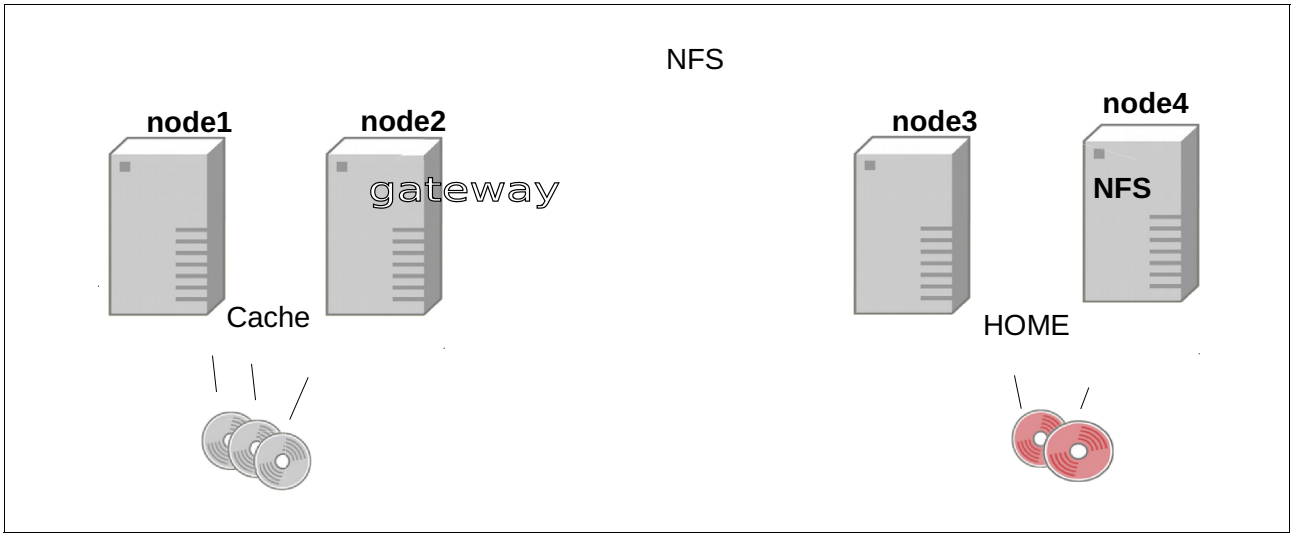


Figure 6-2 Demonstration setup

To start, you need to designate at least one node in your cluster as a gateway node. This is done by using the **mmchnode** command. See Example 6-8. After assigning the *gateway* role to node2, verify your changes with the output from the **mmlscluster** command.

Example 6-8 Creating the gateway node

```
node1:~ # mmchnode --gateway -N node2
Wed Oct  8 22:35:42 CEST 2014: mmchnode: Processing node node2.site
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
node1:~ # mmlscluster
[...] some output is filtered for better overview
 Node  Daemon node name  IP address  Admin node name  Designation
-------------------------------------------------------------------
   1   node1.site        10.0.0.111  node1.site       quorum-manager
   2   node2.site        10.0.0.112  node2.site       gateway
node1:~ #
```

The gateway node is responsible for transferring file data between the Cache and the HOME. Changes to the local Spectrum Scale file system can be generated by any member in the IBM Spectrum Scale cluster, however transferring the data to HOME is the duty of the gateway node only. AIX and Linux nodes can act as gateways.

Gateway nodes support failover, so for high availability it is recommended to have at least two gateway nodes in your Cache cluster. When you have more than one gateway node configured, make sure that the NFS exports from HOME are enabled for all gateway nodes. In this scenario, we begin with just one gateway node. The second gateway node is added later.

The next step is to configure the Cache fileset. A Cache fileset is an independent fileset with AFM properties.

Note: Test the NFS mounts from the gateway nodes to the HOME by using the standard operating system mount command before you create and link your fileset.

Validate that NFS access is working from the gateway nodes to the HOME cluster, then create the Cache fileset. Example 6-9 illustrates the steps required to create a Cache fileset.

Example 6-9 Creating an AFM (SW) fileset

```
node1:~ # df -k                # to show, that the GPFS filesystem on CACHE is mounted
Filesystem     1K-blocks    Used Available Use% Mounted on
[...]
/dev/fs1         5726208  546816   5179392  10% /gpfs/cache

node1:/gpfs/cache # mmlsfileset fs          # before you start, verify fileset configuration
Filesets in file system 'fs1':
Name                    Status    Path
root                    Linked    /gpfs/cache

# now, create the fileset
node1:/gpfs/cache # mmcrfileset fs1 fileset_SW -p afmtarget=node4:/gpfs/fshome/fset001 \
                                                -p afmmode=single-writer --inode-space new

Fileset fileset_SW created with id 1 root inode 131075.

# link the fileset
node1:/gpfs/cache # mmlinkfileset fs1 fileset_SW -J /gpfs/cache/fileset_SW
Fileset fileset_SW linked at /gpfs/cache/fileset_SW
node1:/gpfs/cache #
```

Once the AFM fileset is created and linked, it is ready to use.

6.2.2 Working with AFM filesets (single-writer)

At this point, AFM is still not Active on this fileset because no data has been created or modified in the fileset. When the fileset is untouched, AFM stays in an *Inactive* state. Once you enter the directory tree and issue a file command such as **ls -l**, the AFM fileset is activated. You can monitor the state of the AFM filesets with the **mmafmctl** command.

Furthermore, Example 6-10 shows how to monitor the activity in a fileset. Once the fileset becomes *Active* because you initiated it with the command **ls -l** in the root directory, the value of *numExec* increases. Depending on the validation interval, you can now list the content without increasing the AFM execution counter for a certain time.

Example 6-10 Listing the state of AFM Cache state

```
node1:/gpfs/cache # mmafmctl fs1 getstate ### to verify the state just right after the creating/link the fileset
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
```

```
fileset_SW      nfs://node4/gpfs/fshome/fset001             Inactive
node1:/gpfs/cache #

## now, stat the directory, to active AFM
node1:/gpfs/cache # ls -l
total 1
drwx------ 4 root root  4096 Oct  8 20:38 fileset_SW
dr-xr-xr-x 2 root root 32768 Jan  1  1970 .snapshots
# Now the fileset is active and numExec is 1 (for the ls)
node1:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              1
```

Example 6-11 demonstrates how a Cache fileset in single-writer mode works. First, a new validation to HOME is forced by listing the content of the root directory of the Cache fileset. In Example 6-11, AFM tracks usage counts and reports some statistics to verify that it is working properly. The number on the execution column (numExec) increased as the **ls -l** in the directory was initiated. This is because AFM has to validate the content against the HOME directory. When numExec increases, it means that the data was either accessed for the first time or needs to be revalidated.

Example 6-11 Activate AFM queuing

```
node1:/gpfs/cache # ls -l fileset_SW/
total 96
drwx------ 65535 root root 32768 Oct  9 20:14 .afm
drwx------ 65535 root root 32768 Oct  9 20:14 .ptrash
dr-xr-xr-x     2 root root 32768 Jan  1  1970 .snapshots
node1:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              5
node1:/gpfs/cache #
```

Now a few files are created to illustrate the behavior of a single-writer fileset. When some files are created, the Cache State changes to "*Dirty*" because there are entries that have not been transferred to HOME. When the files are created, the *Queue Length* increases. Then, when the work in the queue has been completed, the *Queue NumExec* column is increased.

How long a change sits in the queue before being executed depends on the speed of the network between Cache and HOME and the value of afmAsyncDelay. See Example 6-12.

Example 6-12 Filling the AFM queue

```
## creating 4 new files in the CACHE fileset (cache side)
node1:/gpfs/cache/fileset_SW # for i in 1 2 3 4 ; do; date >> file$i; done
# verify, that the files were written (on cache)
node1:/gpfs/cache/fileset_SW # ls -l
total 96
drwx------ 65535 root root 32768 Oct  9 20:14 .afm
-rw-r--r--     1 root root    30 Oct  9 20:25 file1
-rw-r--r--     1 root root    30 Oct  9 20:25 file2
-rw-r--r--     1 root root    30 Oct  9 20:25 file3
-rw-r--r--     1 root root    30 Oct  9 20:25 file4
drwx------ 65535 root root 32768 Oct  9 20:14 .ptrash
dr-xr-xr-x     2 root root 32768 Jan  1  1970 .snapshots
```

```
# verify Cache state
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Dirty           node2           8              5

## wait for a little while , up to 30 seconds ....

node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              13
node1:/gpfs/cache/fileset_SW #
```

In Example 6-12 on page 301, the files are written into Spectrum Scale on node1 and the gateway node is node2. This demonstrates that AFM works regardless where the data is written into the file system, and it is transferred by AFM automatically over the gateway node to HOME.

Finally, check that AFM is working properly. In Example 6-13, a file is created in the Cache and the Cache state changes to dirty (display the Cache state by using the **mmafmctl** command). At the same time, in HOME check that the files have arrived. After they are available in HOME, the Cache state changes from dirty to active. Refer to Example 6-13.

Example 6-13 AFM object transferring to HOME

```
# create the file on CACHE
node1:/gpfs/cache/fileset_SW # date >> file8
# verify cache state
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Dirty           node2           2              22
node1:/gpfs/cache/fileset_SW #
node1:/gpfs/cache/fileset_SW # sleep 30
# meanwhile on HOME:
                  node4:/gpfs/fshome/fset001 # ls -l file8; sleep 30 ; ls -l file8
                  ls: cannot access file8: No such file or directory
                  -rw-r--r-- 1 root root 30 Oct  9 22:21 file8
                  node4:/gpfs/fshome/fset001
# now verify chache state again to see that it turned from drity to "active"
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              24
```

6.2.3 Running single-writer fileset in disconnected mode

AFM is designed to work in a disconnected state. Disconnected means that the gateway node cannot mount the HOME export for some reason. To support this scenario, Spectrum Scale automatically maintains the connection (as possible). After a network outage when the connectivity between Cache and HOME is back for use, AFM automatically reconnects and continues processing the queues.

Example 6-14 shows how AFM works in single-writer mode when the network connection to HOME is lost.

Example 6-14 Running AFM in disconnected mode (1/2)

```
###### ifdown eth0 on node4 # to cut the connectivity
# verify the cache state on CACHE immediately after....
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate -j fileset_SW
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              24
```

When HOME is unavailable, the state of the fileset remains active if no access to the fileset is initiated, and AFM has no need to push changes to HOME. In Example 6-15, a stat() on the directory triggers AFM to recognize the loss of the connection. By default, it can take up to 60 seconds to recognize a HOME failure. Refer to 6.5, “Customizing and tuning AFM filesets” on page 335.

Example 6-15 Running AFM in disconnected mode (2/2)

```
node1:/gpfs/cache/fileset_SW # ls -l

### it takes up to 60 seconds according to your setting - Table 6-3 on page 336 ###

total 96
drwx------ 65535 root root 32768 Oct  9 20:14 .afm
-rw-r--r--     1 root root    30 Oct  9 20:25 file1
-rw-r--r--     1 root root    30 Oct  9 20:25 file2
[...]
# now we verify the cache state
node1:/gpfs/cache/fileset_SW #
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate -j fileset_SW
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Disconnected    node2           0              25
```

As you can see in Example 6-16, the fileset is still operable. After the timeout, the fileset state changes to *Disconnected* mode, although you can keep on using the Spectrum Scale file system as usual.

Example 6-16 demonstrates what happens with the changes made while working with this fileset during the disconnected state.

Example 6-16 Changes to Spectrum Scale during AFM disconnected state

```
# to give an example a directory is created
node1:/gpfs/cache/fileset_SW # mkdir created_during_networkDOWN
node1:/gpfs/cache/fileset_SW # ll
total 128
drwx------ 65535 root root 32768 Oct  9 20:14 .afm
drwxr-xr-x     2 root root 32768 Oct  9 23:23 created_during_networkDOWN
-rw-r--r--     1 root root    30 Oct  9 20:25 file1
-[...]
node1:/gpfs/cache/fileset_SW # cd created_during_networkDOWN/

# now copying some files to that directory, to increase Queue Length
node1:/gpfs/cache/fileset_SW/created_during_networkDOWN # ll
total 0
```

```
node1:/gpfs/cache/fileset_SW/created_during_networkDOWN # cp ../file* .
node1:/gpfs/cache/fileset_SW/created_during_networkDOWN # cd ..
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate -j fileset_SW
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Disconnected    node2           15             25
node1:/gpfs/cache/fileset_SW #
```

As you can see in Example 6-17, you can use the file system as usual, even though the state of the fileset remains in disconnected. What happens is that the queue length increases according to the number of changes. AFM recognizes when the connection to HOME is repaired and available again. Then, AFM automatically pushes the content of the queue to HOME. When all of the outstanding work is flushed the state turns to active again.

Example 6-17 Recover from disconnected state

```
node1:/gpfs/cache/fileset_SW # # bring network back
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate -j fileset_SW
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Dirty           node2           15             25
node1:/gpfs/cache/fileset_SW #

node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate -j fileset_SW
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              41
node1:/gpfs/cache/fileset_SW #
```

In comparison to AFM modes such as read-only or local-updates, the single-writer mode checks that existing data in the Cache remains 100% accessible during the loss of connection to HOME. No data expires or is evicted in AFM by default, as long as the mode is single-writer. Data that has been evicted from Cache is not accessible during the time that HOME is disconnected.

The other Cache mode scenarios are discussed later in this chapter.

6.2.4 Auto recovering from failed connections

As mentioned, reconnection to the HOME is done automatically by the gateway. When a fileset in single-writer mode is running for a long time in a disconnected mode, it might be required to be resynced to HOME by the fact that the disconnection might have caused some local or remote outages. Filesets, acting in read-only or local-update mode, work without this resync mechanism because they validate the data periodically. Single-writer and independent-writer work differently.

GPFS recognizes the need for recovery automatically and initiates it in the background. An automated recovery occurs, for example, when you restart your Cache after maintenance purposes. The gateway node recognizes mismatches when activating the fileset and initiates the resync. You get a message in the mmfs log as shown in Example 6-18.

Example 6-18 AFM recovery

```
Sat Oct 11 15:28:15 CEST 2014: [N] AFM: Starting recovery for fileset 'fileset_SW' in fs 'fs1'
Sat Oct 11 15:28:15 CEST 2014: mmcommon afmrecovery invoked: device=fs1 filesetId=1 [....]
```

```
Sat Oct 11 15:28:16 CEST 2014: [N] AFM: mmafmlocal /usr/lpp/mmfs/bin/mmapplypolicy [....]
[...] some output depressed to fit into the screen
Sat Oct 11 15:28:31.325 2014: [I] AFM: Detecting operations to be recovered...
Sat Oct 11 15:28:31.328 2014: [I] AFM: Found 2 update operations...
Sat Oct 11 15:28:31.331 2014: [I] AFM: Starting 'queue' operation for fileset 'fileset_SW' in
filesystem 'fs1'.
Sat Oct 11 15:28:31.332 2014: [I] Command: tspcache fs1 1 fileset_SW 0 3 1346604503 38 0 43
Sat Oct 11 15:28:31.375 2014: [I] Command: successful tspcache fs1 1 fileset_SW 0 3 1346604503
38 0 43
Sat Oct 11 15:28:31 CEST 2014: [I] AFM: Finished queuing recovery operations for
/gpfs/cache/fileset_SW
```

During the recovery time, the fileset (SW) is accessible and ready to use without hampering application on the Cache side. However, the AFM Cache state changes during this time as shown in Example 6-19.

Example 6-19 AFM Cache state during recovery

```
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         FlushOnly       node2           0              0
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Recovery        node2           0              0
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              3
node2:/gpfs/cache/fileset_SW #
```

In rare cases, the resync might fail due to insufficient system resources, further outages, or any other unexpected scenario. The Cache state might change then to Dropped. Then, an administrator has to find out the reason. A good starting point for further investigation is the mmfslog on the gateway nodes. In addition, keep in mind that enough system resources must be available to generate snapshots in the fileset.

6.2.5 Manual recovering from failed connections

When a given fileset gets stuck in NeedsResync state, first you have to find out the root cause, which is mostly related to missing capabilities to sync the data back to AFM HOME. Possible reasons may be related to snapshots, system resources, memory utilization, or any other circumstances that are related to the data on both sides. So it is worth to check also the operating system log facilities for further debugging. NFS-related failures can be reported on HOME only. Once you solved the issue, resynchronize AFM to HOME as shown in Example 6-20.

Example 6-20 Initiate manual resync of AFM (SW)

```
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      NeedsResync     node2           61              0
node2:/gpfs/cache/fileset_SW #
```

```
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 resync -j fileset_SW
mmafmctl: Performing resync of fileset: fileset_SW
node2:/gpfs/cache/fileset_SW #
# the fileset turn to Active after Recovery
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              62
```

If you cannot resync the data permanently because of inconsistencies in HOME, you might solve the situation by referring to 6.2.6, “Permanent loss of HOME for single-writer” on page 306, or you might have to choose another Cache mode for your fileset scenario.

In addition, it is likely that errors on HOME prevent AFM from writing data to HOME. You see an increasing queue counter without sending data to HOME. The file gets stuck in the Dirty state. The mmfs log facilities should be checked as well for problems. As you can see in the following examples, an error is forced on HOME to demonstrate the behavior of Cache. Example 6-21 shows how non-working AFM connections can be discovered on Cache.

Example 6-21 Dirty Cache state

```
node2:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Dirty           node2           14             25(5)
node2:~ #sleep 60
node2:~ #mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Dirty           node2           14             25(5)
```

As you can see in Example 6-21, something prevented AFM from working correctly. The fileset hangs in Dirty mode, and even after 60 seconds, the queue counters do not change. Refer to the mmfslog facilities as shown in Example 6-22.

Example 6-22 Sample mmfs.log entry for failed AFM

```
Sat Oct 11 16:25:33.966 2014: [E] AFM: Remove file system fs1 fileset fileset_SW
file IDs [131075.131338.-1.-1,N] name file111 remote error 13
Sat Oct 11 16:25:33.967 2014: Permission denied
```

After you correct the situation on HOME, resume the requeued entries as shown in Example 6-23. After initiating the command, AFM goes back to normal operations and the queue state should show Active soon.

Example 6-23 Resume requeue for AFM

```
node2:~ # mmafmctl fs1 resumeRequeued -j fileset_SW
node2:~ #
```

6.2.6 Permanent loss of HOME for single-writer

A loss of HOME due to a disaster is not complicated to recover from with AFM single-writer. In one sentence, you point to a new HOME, and AFM repopulates it.

If your HOME file system is permanently lost, you can easily create a new HOME and request AFM to change HOME for the fileset. When you assign a new afmTarget for a single-writer fileset, all files are transferred to the new HOME. When you create a new HOME, it needs to be empty. Once these steps are completed, you can change the Cache fileset configuration accordingly. The single-writer fileset is accessible after changing the HOME. All the data is copied in background. During the repopulation of HOME, the Cache is in a recovery state. To define a new HOME, use the **mmafmctl** command with the failover option. See Example 6-24.

Example 6-24 Failover scenario to a new HOME

```
node2:/gpfs/cache/fileset_SW # mmlsfileset fs1 fileset_SW --afm
# show the origin configuration
#
Filesets in file system 'fs1':
Name                    Status    Path                                    afmTarget
fileset_SW              Linked    /gpfs/cache/fileset_SW                  nfs://node4/gpfs/fshome/fset001
#now changing
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 failover -j fileset_SW --new-target nfs://node4/gpfs/fshome/fset002new
mmafmctl: Performing failover to nfs://node4/gpfs/fshome/fset002new
Fileset fileset_SW changed.
mmafmctl: Failover in progress. This may take while...
 Check fileset state or register for callback to know the completion status.
node2:/gpfs/cache/fileset_SW #
node2:/gpfs/cache/fileset_SW # mmlsfileset fs1 --afm
Filesets in file system 'fs1':
Name                    Status    Path                                    afmTarget
root                    Linked    /gpfs/cache                             --
fileset_SW              Linked    /gpfs/cache/fileset_SW                  nfs://node4/gpfs/fshome/fset002new
node2:/gpfs/cache/fileset_SW #
```

During this time, the AFM state changes as shown in Example 6-25.

Example 6-25 Failover scenario to a new HOME - Cache states

```
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset001         Active          node2           0              6
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Inactive
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      NeedsResync     node2           6              0
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Recovery        node2           6              0
node2:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              6
```

6.2.7 Permanent loss of Cache (single-writer)

If you are running AFM single-writer for disaster recovery (DR) purposes when everything is working properly, the production application is accessing the single-writer Cache so that any changes to the data are transferred to HOME. When the Cache fails, an administrator or any other mechanism has to redirect the production workload to use HOME (AFM cannot redirect a production workload; this is done by some other method) to continue operations. Keep in mind that the content on HOME might not be 100% identical to the Cache fileset. The state of the data depends on the asyncDelay setting and network capabilities between Cache and HOME. Figure 6-3 illustrates the scenario.

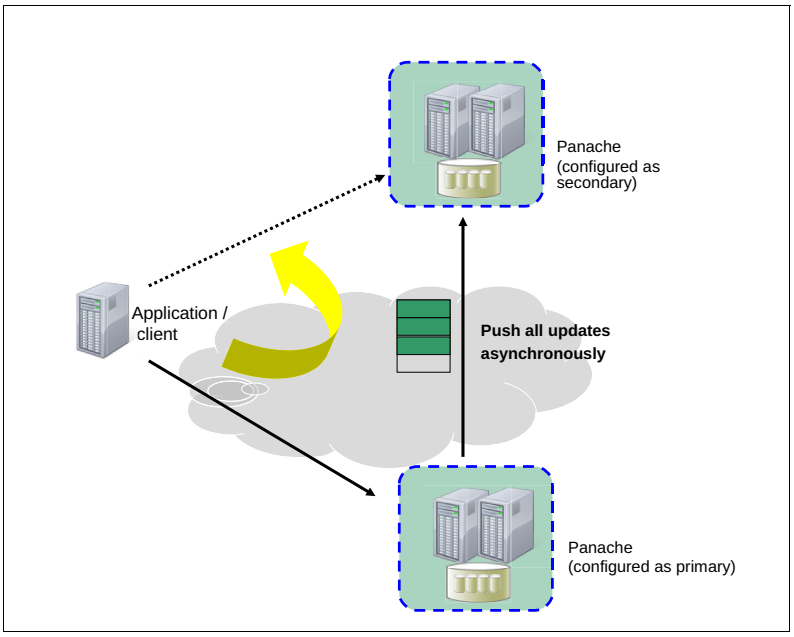


Figure 6-3 Disaster recovery: Permanent loss of Cache

Even if you are able to recover your Cache fileset, you must not reactivate your single-writer Cache once data on HOME has changed. Changes on HOME for a single-writer are not supported and might lead to loss of data, inconsistencies, and stopping AFM working correctly.

Therefore, you need to rebuild your Cache fileset from scratch, which means that all data needs to be copied from HOME to the newly created Cache. Although the copy of data might take some time, even some days, all data becomes visible and accessible on the Cache side immediately after the configuration is done. Accessed data, which is not yet synchronized in Cache, is fetched immediately on access and therefore some performance penalty might occur. To avoid this, you can use prefetch policies to prefetch important data first. As a matter of fact, all data is available on Cache for application access.

Following are the basics steps for this procedure:

1. Move the production workload to HOME.
2. If the failed fileset still exists, unlink the existing, non-valid Cache fileset. Delete the fileset including the **-f** option for removing the files as well.
3. Create a fileset and switch back.
4. Verify that any workload on remote HOME is stopped and the network to HOME is available.
5. Link the new fileset and start using it.

For an example on the preceding steps, refer to Example 6-26, which shows the content of a given SW Cache fileset before the disaster scenario. The state of the AFM Cache has turned to *Disconnected*.

Example 6-26 Verify state before recovering from loss of Cache

```
node1:/gpfs/cache/fileset_SW # ls -lrt ######### to demonstrate, that files won't be lost after these steps
total 65
drwx------ 65535 root root 32768 Oct  9 20:14 .afm
dr-xr-xr-x     2 root root 32768 Oct 10 20:51 .snapshots
drwx------ 65535 root root 32768 Oct 13 16:11 .ptrash
-rw-r--r--     1 root root    30 Oct 13 16:19 see
-rw-r--r--     1 root root    30 Oct 13 16:19 my
-rw-r--r--     1 root root    30 Oct 13 16:19 files
-rw-r--r--     1 root root    30 Oct 13 16:19 from
-rw-r--r--     1 root root    30 Oct 13 16:19 yesterday
node1:/gpfs/cache/fileset_SW # cd ..
node1:/gpfs/cache # mmlsfileset fs1
Filesets in file system 'fs1':
Name                    Status    Path
root                    Linked    /gpfs/cache
fileset_SW              Linked    /gpfs/cache/fileset_SW                # the fileset is still linked

node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new        Disconnected    node2           0              32
node1:/gpfs/cache/fileset_SW # halt -q # all nodes in cluster were halted to pretend side outage now
```

The state is already *disconnected*, so no transfer of any data to HOME is possible. Assuming that Cache is lost due to a hardware error or power failures, these verifications might not be possible. It is also possible to use this procedure for longer maintenance windows.

When an outage occurs, the application workload can be pointed to HOME. Data is accessible as illustrated Figure 6-3 on page 308.

Note: Since data on HOME is changing, the content of the Cache fileset in single-writer mode becomes invalid. The SW fileset needs to be recycled.

When the origin Cache side is back and ready for operations, we continue with command two and command three as shown in Example 6-27.

Example 6-27 Command two and command three, unlink and delete origin cache fileset

```
node1:/gpfs/cache # mmlsfileset fs1 --afm
Filesets in file system 'fs1':
Name                    Status    Path                                  afmTarget
root                    Linked    /gpfs/cache                           --
fileset_SW              Linked    /gpfs/cache/fileset_SW                nfs://node4/gpfs/fshome/fset002new
nnode1:/gpfs/cache # mmunlinkfileset fs1 fileset_SW -f
Fileset fileset_SW unlinked.
node1:/gpfs/cache # mmdelfileset fs1 fileset_SW -f
Checking fileset ...
Checking fileset complete.
Deleting user files ...
 100.00 % complete on Mon Oct 13 16:34:54 2014  (     66560 inodes with total        260 MB data processed)
Deleting fileset ...
```

```
Fileset fileset_SW deleted.
```

After the fileset is deleted, which can take a while depending on the number of objects in the fileset, proceed with step 4 (Example 6-28) and create a new Cache fileset pointing to the existing data on HOME. During this time, HOME is operable and still can be used and data on HOME can change as well as shown in Example 6-28.

Example 6-28 Step 4: Create a fileset

```
node1:/gpfs/cache #
node1:/gpfs/cache # mmcrfileset fs1 fileset_SW -p afmtarget=nfs://node4/gpfs/fshome/fset002new -p afmmode=sw --inode-space=new
Fileset fileset_SW created with id 1 root inode 131075.
```

Now, before the new fileset is linked in the Cache and becomes accessible, the application load on HOME must be suppressed. If it is confirmed that data on HOME has not changed anymore, you can proceed to step 6 as shown in Example 6-29.

Example 6-29 Activating the new cache SW fileset

```
node1:/gpfs/cache # mmlinkfileset fs1 fileset_SW # link the fileset
Fileset fileset_SW linked at /gpfs/cache/fileset_SW #

######## verify state change of Cache fileset
node1:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Inactive
node1:/gpfs/cache # cd fileset_SW/ # now you active the fileset, by accessing the DIR
node1:/gpfs/cache/fileset_SW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              13

# to demonstrate, that the origin files are there again .. and some new
node1:/gpfs/cache/fileset_SW # ls -lrt #
total 96
-rw-r--r--     1 root root    30 Oct 13 16:19 see
-rw-r--r--     1 root root    30 Oct 13 16:19 my
-rw-r--r--     1 root root    30 Oct 13 16:19 files
-rw-r--r--     1 root root    30 Oct 13 16:19 from
-rw-r--r--     1 root root    30 Oct 13 16:19 yesterday
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
-rw-r--r--     1 root root    30 Oct 13 16:32 +++++
-rw-r--r--     1 root root    30 Oct 13 16:32 created
-rw-r--r--     1 root root    30 Oct 13 16:32 during
-rw-r--r--     1 root root    30 Oct 13 16:32 outage
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
drwx------ 65535 root root 32768 Oct 13 16:50 .ptrash
node1:/gpfs/cache/fileset_SW #
```

6.2.8 Summary for AFM with single-writer

Using single-writer, you can build very efficient disaster recovery scenarios. The main advantage to Spectrum Scale internal replication is that the data is duplicated independently from the local Spectrum Scale, which is essential in some use cases. In addition, the changes

to the data are maintained continuously in the background and this makes this solution independent from the total number of objects in a given name space or file system tree.

6.3 AFM independent-writer

This section describes the AFM independent-writer.

6.3.1 Steps to create AFM enabled fileset in independent-writer mode

Operating a Cache fileset in independent-writer mode is similar to single-writer. This section covers the differences between single-writer and independent-writer. The fundamental difference between single-writer and independent-writer is that independent-writer checks HOME for changes, single-writer does not.

Independent-writer works even when data on HOME changes. Since you can have multiple places where updates can occur, be careful in how you use independent-writer. For example, data in the same directories should not be changed by Cache and HOME at the same time. However, AFM in independent-writer mode is capable of handling circumstances like this. In general, it behaves like any other shared file space, and the last updates can override earlier changes.

Tip: Try to streamline your workload from a different independent-writer fileset to HOME, so that they are not writing to the same directory on HOME. Try to separate your workloads instead into a corresponding directory structure in HOME.

Independent-writer Cache behaviors:

1. If a file exists in the Cache (new file) but not on the HOME, the file is copied to the HOME.
2. If an existing file is accessed in the Cache, and a newer version of the file exists in HOME, the file in Cache is updated to the HOME version before it is accessed.
3. If a file in the Cache is older than the version at HOME, the Cache version is discarded and the HOME version is retrieved on the next access from the Cache.

Independent-writer mode can be used to implement a data replication environment, which is capable of changing the data on both sides. One difference compared to single-writer is that in IW mode, the content in the Cache fileset has to be validated periodically. This validation can add additional network traffic. Validation ensures that the Cache is aware of changes on HOME to maintain consistency.

You can create a fileset from scratch, or you can convert an existing fileset from another mode to independent-writer. To change a fileset to a different mode, you need to unlink the fileset or unmount the file system.

Example 6-30 demonstrates how to change an independent-writer fileset to a single-writer.

Example 6-30 Change fileset AFM mode (1/2)

```
node1:~ # mmlsfileset fs1 --afm
Filesets in file system 'fs1':
Name                    Status    Path                                    afmTarget
root                    Linked    /gpfs/cache                             --
fileset_SW              Linked    /gpfs/cache/fileset_SW                  nfs://node4/gpfs/fshome/fset002new
node1:~ # mmumount fs1 -a
Wed Oct 22 05:17:00 CEST 2014: mmumount: Unmounting file systems ...
```

```
node1:~ # mmchfileset fs1 fileset_SW -p afmmode=iw
Fileset fileset_SW changed.
node1:~ # mmlsfileset fs1 fileset_SW --afm -L | grep Mode
Mode                                 independent-writer
node1:~ # mmmount fs1 -a
Wed Oct 22 05:17:38 CEST 2014: mmmount: Mounting file systems ...
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new        Inactive
node1:~ # ls -l /gpfs/cache/fileset_SW/ > /tmp/dummy
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new        Active          node2           0              12
node1:~ #
```

The fileset is now running in independent mode. After accessing the directory with the command **`ls -l`**, it becomes in *Active* state as expected.

You can change back to single-writer. See Example 6-31. Finally, we change it back into IW mode to proceed with our examples.

Example 6-31 Change fileset AFM mode (2/2)

```
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_SW      nfs://node4/gpfs/fshome/fset002new        Active          node2           0              12
node1:~ #

# revert it to single writer
node1:~ # mmunlinkfileset fs1 fileset_SW
Fileset fileset_SW unlinked.
node1:~ # mmchfileset fs1 fileset_SW -p afmmode=sw
Fileset fileset_SW changed.
node1:~ # mmlinkfileset fs1 fileset_SW -J /gpfs/cache/fileset_SW
Fileset fileset_SW linked at /gpfs/cache/fileset_SW
node1:~ # mmlsfileset fs1 fileset_SW --afm -L | grep Mode
Mode                                 single-writer
# but we now change it back for further use with independent writer
node1:~ # mmunlinkfileset fs1 fileset_SW
Fileset fileset_SW unlinked.
node1:~ # mmchfileset fs1 fileset_SW -p afmmode=iw
Fileset fileset_SW changed.
node1:~ # mmlinkfileset fs1 fileset_SW -J /gpfs/cache/fileset_SW
Fileset fileset_SW linked at /gpfs/cache/fileset_SW
node1:~ # mmlsfileset fs1 fileset_SW --afm -L | grep Mode
Mode                                 independent-writer
node1:~ #
```

The final step in our example is to adjust the name accordingly to avoid confusion. To change the fileset name, follow and verify Example 6-32.

Example 6-32 Change fileset name

```
node1:~ # mmchfileset fs1 fileset_SW -j fileset_IW
Fileset fileset_SW changed.
node1:~ # mmunlinkfileset fs1 fileset_IW ; mmlinkfileset fs1 fileset_IW -J /gpfs/cache/fileset_IW
Fileset fileset_IW was unlinked.
```

```
Fileset fileset_IW linked at /gpfs/cache/fileset_IW
node1:~ # mmlsfileset fs1
Filesets in file system 'fs1':
Name                    Status    Path
root                    Linked    /gpfs/cache
fileset_IW              Linked    /gpfs/cache/fileset_IW
node1:~ #
```

To create an independent-writer fileset from scratch, you can use the same command as shown in Example 6-9 on page 300. Use the **mmcrfileset** command but change the attribute afmmode to independent-writer (IW) (short form).

6.3.2 Working with independent-writer filesets

Independent filesets have two hidden directories where AFM stores files that encountered a conflict when an update occurs. To demonstrate how changes on HOME affect the Cache, all the files are deleted in HOME and disappeared in Cache.

The first step is to verify that the fileset is in active state as shown in Example 6-33.

Example 6-33 Verify AFM Cache state

```
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              12
node1:/gpfs/cache/fileset_IW #
```

Now all the files are deleted in HOME. Next, on the Cache side the associated directory shows that the files have been deleted as shown in Example 6-34.

Example 6-34 Listing directory content inside AFM after changes on HOME

```
# on HOME
node4:/gpfs/fshome/fset002new # ll
total 1
-rw-r--r-- 1 root root    30 Oct 13 16:32 +++++
drwxr-xr-x 3 root root  4096 Oct 11 16:05 .afm
-rw-r--r-- 1 root root    30 Oct 13 16:32 created
-rw-r--r-- 1 root root    30 Oct 13 16:32 during
-rw-r--r-- 1 root root    30 Oct 13 16:19 files
-rw-r--r-- 1 root root    30 Oct 13 16:19 yesterday
node4:/gpfs/fshome/fset002new # rm *
node4:/gpfs/fshome/fset002new #

# on Cache it looks like
node1:/gpfs/cache/fileset_IW # ll
ls: cannot access files: No such file or directory
ls: cannot access yesterday: No such file or directory
ls: cannot access +++++: No such file or directory
ls: cannot access during: No such file or directory
ls: cannot access created: No such file or directory
total 128
-????????     ? ?    ?        ?            ? +++++
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
-????????     ? ?    ?        ?            ? created
-????????     ? ?    ?        ?            ? during
```

```
-????????     ? ?    ?        ?            ? files
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 15:42 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
-????????     ? ?    ?        ?            ? yesterday
```

In the output, there are question marks “?” where you might expect to see other information, occasionally listing the directory content immediately might look a bit strange. This is expected as shown in Example 6-34 on page 313 when changes in HOME were initiated right before.

A few seconds later, verifying the changes on Cache again show that AFM has successfully taken the changes from HOME. Example 6-35 shows that all files were cleanly deleted in Cache.

Example 6-35 AFM in independent mode

```
node1:/gpfs/cache/fileset_IW # ll
total 128
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 15:42 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # l
```

The AFM hidden directories `.afm`, `.pconflicts`, and `.ptrash` must not be removed. These are directories that are used by AFM to maintain the data set. Once created, these directories remain even if the Cache mode is changed. A file can occasionally end up in the `.ptrash` directory on the Cache side if they had been deleted on HOME. So you should develop house-keeping plans for this data; for example, keep conflicts for some number of days.

Tip: Files moved back from `.ptrash` or `.pconflicts` are treated as local files to the Cache fileset and are not queued to HOME.

To show this behavior, a file is created in the Cache, which is verified to be copied HOME, and then we delete the file on HOME. AFM works as expected and places the file in `.ptrash` as shown in Example 6-36.

Example 6-36 AFM special directories

```
node1:/gpfs/cache/fileset_IW # echo "hallo, created on Cache" > newfile1
node1:/gpfs/cache/fileset_IW # ls -l
total 128
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
-rw-r--r--     1 root root    24 Oct 22 16:41 newfile1
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 16:38 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new        Dirty           node2           3              65
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new        Active          node2           0              68
node1:/gpfs/cache/fileset_IW # ssh node4 "ls -l /gpfs/fshome/fset002new/newfile1"
```

```
-rw-r--r-- 1 root root 24 Oct 22 16:41 /gpfs/fshome/fset002new/newfile1
node1:/gpfs/cache/fileset_IW #

# Now we're going to remove the file remotely on HOME and verify the changes in cache
node1:/gpfs/cache/fileset_IW # ssh node4 "rm /gpfs/fshome/fset002new/newfile1"
node1:/gpfs/cache/fileset_IW # ls -l
total 128
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 16:44 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # ls -l .ptrash
total 0
-rw-r--r-- 1 root root 24 Oct 22 16:41 newfile1.1413989076.585354669
node1:/gpfs/cache/fileset_IW #
```

To demonstrate the behavior of AFM when moving files back to the original place, watch the queue state, and you see that this file is not copied to HOME again as shown in Example 6-37.

Example 6-37 Special file treatment

```
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                              Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                              -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new          Active          node2           0              73
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate | tail -1 | awk '{print $5}'
0
node1:/gpfs/cache/fileset_IW # ls -l .ptrash/newfile1.1413989076.585354669
-rw-r--r-- 1 root root 24 Oct 22 16:41 .ptrash/newfile1.1413989076.585354669
node1:/gpfs/cache/fileset_IW # mv .ptrash/newfile1.1413989076.585354669 ./newfile1
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                              Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                              -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new          Active          node2           0              74

node1:/gpfs/cache/fileset_IW # ls -l
total 128
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
-rw-r--r--     1 root root    24 Oct 22 16:41 newfile1
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 16:47 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # sleep 60; ssh node4 "ls -l /gpfs/fshome/fset002new/newfile1"
ls: cannot access /gpfs/fshome/fset002new/newfile1: No such file or directory
node1:/gpfs/cache/fileset_IW #
```

If you want the file to act as any other, copy it back to the fileset (instead of using the **mv** command) and delete it from the `.ptrash` directory. Then, AFM is aware of the file and transfers it to HOME.

Note: Files from `.ptrash` must be copied back to its original location to continue to be managed by AFM, which copies them back to their original place.

The following summarizes the behavior of an independent-writer fileset:

- An independent-writer fileset is aware of file changes in HOME.
- If a conflict is detected during an asynchronous data operation:

- Use existing file in HOME and send updates to the existing file.
- If that is not possible (for example, the existing file is not of the same type), ignore the error (but mark the file as a problem) and handle during revalidation.
- On revalidation, if inode changes exist, delete non-dirty files and backup dirty files or directories in Cache to the `.ptrash` directory.
- No checks when executing async write, data simply flows to the HOME file.

6.3.3 Cache eviction in independent-writer mode

AFM implements a per-fileset Cache for a remote file system and so a Spectrum Scale fileset can be configured to serve as an on-disk Spectrum Scale Cache for remote data. Presently, objects (files or directories) are Cached in their entirety as part of a user-initiated read operation, when AFM discovers that the file or directory is “incomplete”.

AFM maintains file attribute consistency between local and remote data. However, certain attributes like inode number and file times can be different. You can use an independent-writer Cache that has less storage than HOME. When running in this configuration, the Cache might need to be pruned periodically. Removing file data from a Cache is called *evicting a file*. When a file is evicted, the file data is removed from the Cache and the inode stays in the Cache. This is done so local inode numbers that might have been exposed to applications are not changed by recaching the inode.

Using eviction allows you to build environments, where all objects from HOME are available but running in a very small, limited amount of space, for example this Cache could be created in flash storage. This opens a very powerful method of providing small but high speed and low latency Caches to clusters.

Continuing with our example, demonstrate how Cache eviction can be used. Start with a very small limited amount of free space in the Cache, then free space using eviction. When a file is evicted, it does not occupy space in Cache besides the inode. A very small Cache can hold all of the metadata from HOME and therefore every object from HOME can be maintained in the Cache as well. Refer to Figure 6-4.

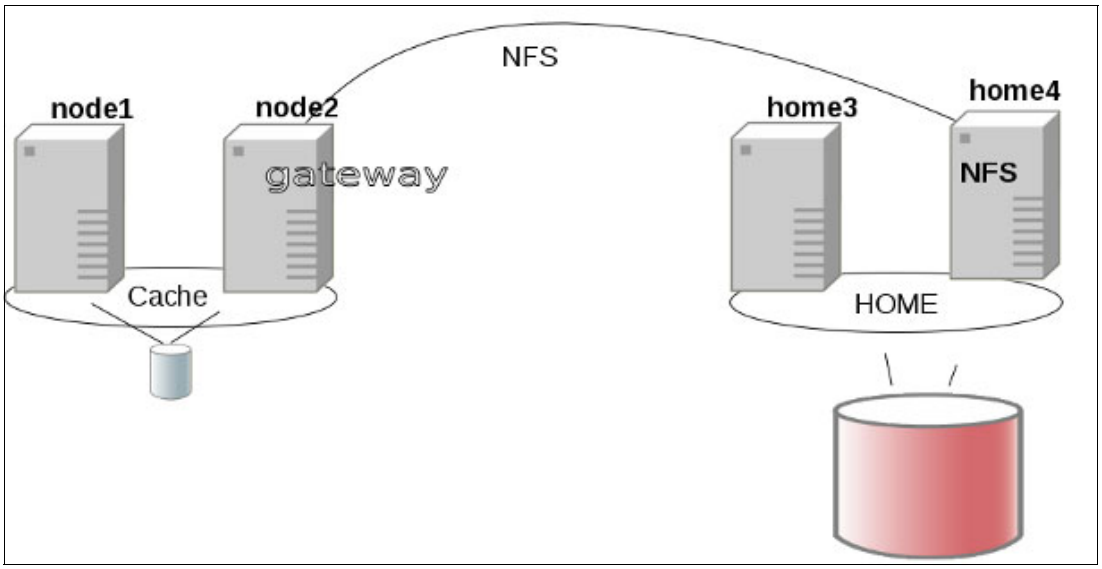


Figure 6-4 Cache eviction scenario

Example 6-38 creates several “write*” files and some “work*” files until the Cache is full. Then, it demonstrates the steps that it takes to evict files.

Example 6-38 No space is left on Cache

```
node1:/gpfs/cache/fileset_IW # ls -l
total 2380129
drwxr-xr-x     5 root root       4096 Oct 29 08:43 ./
drwxr-xr-x     3 root root     262144 Oct 24 16:35 ../
drwx------ 65535 root root      32768 Oct 24 01:01 .afm/
drwx------ 65535 root root      32768 Oct 24 01:01 .pconflicts/
drwx------ 65535 root root      32768 Oct 24 01:42 .ptrash/
dr-xr-xr-x     2 root root      32768 Jan  1  1970 .snapshots/
-rw-r--r--     1 root root  104857600 Oct 29 08:30 workfile1
[...]
-rw-r--r--     1 root root  104857600 Oct 29 08:30 workfile5
-rw-r--r--     1 root root  104857600 Oct 24 16:40 writefile1
-rw-r--r--     1 root root  104857600 Oct 24 16:46 writefile2
[...]
-rw-r--r--     1 root root  104857600 Oct 29 08:17 writefile8
node1:/gpfs/cache/fileset_IW # df -k
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda2        8811252 4541508   3822156  55% /
devtmpfs          218688     132    218556   1% /dev
tmpfs             218688       0    218688   0% /dev/shm
/dev/fs1         3147776 3147776         0 100% /gpfs/cache
node1:/gpfs/cache/fileset_IW # dd if=/dev/zero bs=1M count=1000 of=workfile.next
dd: writing `workfile.next': No space left on device
[...]
```

In Example 6-38, “No space left on device” indicates that no more writes are possible. The existing files in that directory, named as `workfile` or `writefile`, are needed soon.

To use Cache eviction, the file system quotas need to be enabled. See 5.6, “Quotas” on page 267 for general considerations about quotas.

Note: Quota enforcement needs to be turned on to use Cache eviction.

The next step is to verify that the file system has quotas enabled as shown in Example 6-39. If quotas are not enabled, use the **mmquotaon** command to enable quotas.

Example 6-39 Enable quota for AFM Cache eviction (2/2)

```
node1:/gpfs/cache/fileset_IW # mmlsfs fs1 -Q
flag                value                    description
------------------- ------------------------ -----------------------------------
 -Q                 user;group;fileset       Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
node1:/gpfs/cache/fileset_IW #

node1:/gpfs/cache/fileset_IW # mmquotaon -v fs1
mmquotaon on fs1
node1:/gpfs/cache/fileset_IW # mmlsfs fs1 -Q
flag                value                    description
------------------- ------------------------ -----------------------------------
 -Q                 user;group;fileset       Quotas accounting enabled
                    user;group;fileset       Quotas enforced
```

```
                  none                    Default quotas enabled
node1:/gpfs/cache/fileset_IW #
```

In addition to enabling quotas, at least one default quota must be configured. If you have never used quotas in this fileset before, you need to run the **mmcheckquota** command. If quotas are not ready for use, you get an error message as shown in Example 6-40.

Example 6-40 Enable quota for AFM Cache eviction (1/2)

```
node1:/gpfs/cache/fileset_IW # mmlsquota -j fileset_IW fs1
                       Block Limits                                    |     File Limits
Filesystem type             KB      quota      limit   in_doubt    grace |    files   quota    limit in_doubt    grace  Remarks
fs1        FILESET          96     102400    2097152          0     none |       20       0        0        0     none

node1:/gpfs/cache/fileset_IW # mmafmctl fs1 evict -j fileset_IW --filter FILENAME='%work%'
mmafmctl: Run mmcheckquota command before running mmafmctl with evict option
mmafmctl: Command failed. Examine previous error messages to determine cause.
node1:/gpfs/cache/fileset_IW # mmcheckquota fs1
fs1: Start quota check
   2 % complete on Wed Oct 29 09:19:41 2014
 […]
  95 % complete on Wed Oct 29 09:20:03 2014
 100 % complete on Wed Oct 29 09:20:03 2014
Finished scanning the inodes for fs1.
Merging results from scan.
node1:/gpfs/cache/fileset_IW #
```

Now the fileset is ready and you can evict files. You can use lists that are generated by the Spectrum Scale policy engine or provide the list through some other method. When running the **mmafmctl eviction** command, you can specify which method to use to determine which files should be evicted. These options include ranking the selection according to least recently used (LRU) or the size of the files or limiting the selection to files greater than or smaller than a specified value. Example 6-41 shows the use of the FILENAME filter to select all files with the word “work” in the name. A ‘%’ (percent) is a wildcard in the Spectrum Scale policy rule syntax. Before evicting the files, check that the files are present in the Cache by using the **mmafmlocal** command.

Example 6-41 AFM Cache eviction

```
node1:/gpfs/cache/fileset_IW # mmafmlocal ls workfile1
-rw-r--r-- 1 root root 104857600 Oct 29 08:30 workfile1
node1:/gpfs/cache/fileset_IW #

node1:/gpfs/cache/fileset_IW # mmafmctl fs1 evict -j fileset_IW --filter FILENAME='%work%'
node1:/gpfs/cache/fileset_IW # mmafmlocal ls workfile1
mmafmlocal: Command failed. Examine previous error messages to determine cause.
node1:/gpfs/cache/fileset_IW #
```

Now that all of the “work” files have been evicted from Cache, there is free space available again. The file inodes are still there. You can see this by using the **stat()** command to display the file attributes. If no data is read from these files, only the metadata is present and no data space is occupied. Once the file is read again, all data is copied into the Cache by AFM and the file uses data space again. See Example 6-42.

Example 6-42 AFM Cache eviction and recall

```
node1:/gpfs/cache/fileset_IW # df -k .
Filesystem     1K-blocks   Used Available Use% Mounted on
/dev/fs1         3147776 761856   2385920  25% /gpfs/cache
```

```
node1:/gpfs/cache/fileset_IW # mmafmlocal ls workfile1
mmafmlocal: Command failed. Examine previous error messages to determine cause.
node1:/gpfs/cache/fileset_IW # ls workfile1
workfile1
node1:/gpfs/cache/fileset_IW # cp workfile1 /dev/null
node1:/gpfs/cache/fileset_IW # mmafmlocal ls workfile1
-rw-r--r-- 1 root root 104857600 Oct 29 08:30 workfile1
node1:/gpfs/cache/fileset_IW #
```

6.3.4 Running independent-writer fileset in disconnected mode

Even when a disconnected state is recognized by the fileset, you can keep using the fileset. However, without a connection to HOME, changes in HOME cannot be recognized by AFM. Keep this in mind if you write to the same data set at the HOME and the Cache regularly as it can lead to conflicts later when the connection is restored. If all of the updates are done in the Cache or the HOME and Cache updates are to distinct data sets, there are no issues.

Example 6-43 shows the creations of five files on each side while the fileset is in disconnected state.

Example 6-43 Independent-writer in disconnected mode

```
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Disconnected    node2           0              102
node1:/gpfs/cache/fileset_IW # ll
total 128 ############### you can still access the files, which were already in cache
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
-rw-r--r--     1 root root     8 Oct 22 17:02 newfile2
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 17:02 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # for i in 1 2 3 4 5 ; do ; date > file$i ; done
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Disconnected    node2           27             102

### now reestablish connection to NFS Server (AFM HOME)

node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dirty           node2           27             102
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dirty           node2           27             102
node1:/gpfs/cache/fileset_IW # date > file10
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              134
node1:/gpfs/cache/fileset_IW #
```

When the connection is reestablished, the state changes to dirty. However, it can take a while until every update is transferred to or copied from HOME. How long it takes to “catch up” depends on the speed of your network and how many changes are outstanding.

After all updates are processed, the Cache state changes back to Active. Listing the directory’s content shows that all files, regardless of whether they were created in HOME or in the Cache, are accessible after a short time (see Example 6-44).

Example 6-44 Directory content after reconnect

```
node1:/gpfs/cache/fileset_IW # ll
total 128
drwx------ 65535 root root 32768 Oct 13 16:50 .afm
-rw-r--r--     1 root root    30 Oct 22 17:17 file1
-rw-r--r--     1 root root    30 Oct 22 17:19 file10
-rw-r--r--     1 root root    30 Oct 22 17:17 file2
-rw-r--r--     1 root root    30 Oct 22 17:17 file3
-rw-r--r--     1 root root    30 Oct 22 17:17 file4
-rw-r--r--     1 root root    30 Oct 22 17:17 homecreatedfile1
-rw-r--r--     1 root root    30 Oct 22 17:17 homecreatedfile2
-rw-r--r--     1 root root    30 Oct 22 17:17 homecreatedfile3
-rw-r--r--     1 root root    30 Oct 22 17:17 homecreatedfile4
-rw-r--r--     1 root root    30 Oct 22 17:17 homecreatedfile5
drwx------ 65535 root root 32768 Oct 22 05:23 .pconflicts
drwx------ 65535 root root 32768 Oct 22 17:02 .ptrash
dr-xr-xr-x     2 root root 32768 Oct 13 16:20 .snapshots
node1:/gpfs/cache/fileset_IW # ll .pconflicts
total 0
node1:/gpfs/cache/fileset_IW #
```

It is important to note that if the Cache cluster is disconnected for an extended period, the number of file system updates may exceed the AFM memory (afmHardMemThreshold) capacity of the gateway nodes. In this situation, all changes continue to be “logged” in stable storage. Upon reconnection, AFM reads the changes from disk and synchronizes its local updates with HOME.

Running in the UNMOUNTED state is slightly different but behaves in a similar manner. The Cache state turns out that the TCP connection to HOME is available, but not the protocol (NFS/GPFS) for accessing the files. When the issue on HOME is fixed and the AFM transport protocol is based on NFS, the state turns back to *Active* after 300 seconds. If running Spectrum Scale as the transport protocol (see 6.7.2, “Native GPFS protocol” on page 339), the fileset needs to be remounted by the administrator on the Cache side to get back the Cache state to *Active*.

6.3.5 Manual recovery from failed connections

Manual recovery is needed when AFM cannot bring back the fileset state to Active. To start, you need to find the root cause, which is typically outside of AFM. Check system logs, mmfslogs on remote and local nodes to help troubleshoot the issue. Since the gateway nodes process all the AFM requests, some messages only appear in the log files on the gateway nodes. In some rare cases, your fileset might be stuck in Dropped state as shown in Example 6-45.

Example 6-45 Dropped state

```
node1:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                             Cache State     Gateway Node    Queue Length   Queue numExec
```

```
------------    -------------                           -------------   ------------    ------------    -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dropped         node2           0               6
```

You might find further hints and error messages in the gateway node in the `mmfs.log.latest` file as shown in Example 6-46.

Example 6-46 mmfs.log.latest for a dropped queue state

```
Wed Oct 22 22:53:06 CEST 2014: [N] AFM: Running following policy scan to find missing updates from /gpfs/cache/fileset_IW
Wed Oct 22 22:53:06 CEST 2014: [N] AFM: mmafmlocal /usr/lpp/mmfs/bin/mmapplypolicy
/gpfs/cache/fileset_IW/.snapshots/fileset_IW.afm.17315 --scope fileset -S fileset_IW.afm.17315 -L 0 -I defer -f
/var/mmfs/afm/fs1-1/recovery/policylist.data -N mount -g /gpfs/cache/fileset_IW/.ptrash --iscan-by-number -P
/var/mmfs/afm/fs1-1/recovery/recoveryPolicy >/var/mmfs/afm/fs1-1/recovery/policyLog 2>&1
AFM: Failed with error 1 to run mmapplypolicy to get the recoveryPolicy file lists, retrying with debug option
Wed Oct 22 22:53:06.796 2014: [E] AFM: Recovery on file system fs1 fileset fileset_IW failed with error 1. Recovery will be
retried on next access after recovery retry interval (120 seconds) or manually resolve known problems and recover the fileset.
```

Often these kinds of issues are related to snapshots issues, related to memory constraints, or incorrect export attributes on HOME. After the issue is resolved, the AFM state returns to Active automatically.

There are situations where the Cache state remains as Dirty. In a Dirty state, you are able to use the Cache as usual, though no files are transferred to HOME. This happens when the NFS (HOME) export is not working with correct permissions or it is exported in read-only mode, which might happen due to maintenance tasks or another administrative change in HOME.

When this happens, you may find error messages in the mmfs.log as shown in Example 6-47.

Example 6-47 Common error messages for failing AFM scenarios

```
Wed Oct 22 23:03:47.873 2014: [E] AFM: Create file system fs1 fileset fileset_IW file IDs [131075.147203.-1.-1,N] name cache4
remote error 30
```

Because of Spectrum Scale in Cache works silently in the background, it is important to monitor Spectrum Scale logs accordingly to be aware of issues. To monitor the performance of data movement, you can monitor the queue length, use a callback to monitor fileset state, or periodically time a **`mmfsadm flushpending`** command to get an idea of the time that it takes to flush the queue. Use the **`mmafmctl`** command to see the queue length and fileset state as shown in Example 6-48.

Note: Monitor queue length and screen mmfs logs for [E]rror messages related to AFM.

Example 6-48 Monitoring queue state and queue length

```
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dirty           node2           2              111(1)
node2:/gpfs/cache/fileset_IW #
node2:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate ; touch newfile1 ; mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dirty           node2           5              235(2)
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
```

```
fileset_IW      nfs://node4/gpfs/fshome/fset002new           Dirty          node2          6             237(2)
node2:/gpfs/cache/fileset_IW #
```

Once you find the root case preventing AFM from writing to HOME, AFM should work correctly. Occasionally the Cache stays in a dirty state. This is usually because there are some entries labeled as requeued as shown in Example 6-49. To restore the Cache to Active state, use the **mmafmctl resume** command (Example 6-49).

Example 6-49 Requeued entries in AFM

```
mmfsadm dump afm | grep requeued
[..]
0 Truncate [-1.147206] requeued etype normal
16 Create [131075.141576] requeued etype normal chafile2222
[...]

node2:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new        Dirty           node2           5              239(2)
node2:~ # mmafmctl fs1 resumeRequeued -j fileset_IW #  <<<<<<<<<<<<<<<<<<<<<<<<< this changes the Cache state back to Active
node2:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new        Active          node2           0              244
node2:~ #
```

6.3.6 Permanent loss of HOME

When HOME is permanently lost, you need to switch the AFM target to a new HOME. This is done by using the **mmafmctl failover** command. When you issue the failover command, AFM queues all data in the fileset to be flushed to the new HOME. For this reason, it is recommended that the new HOME should be empty and the fileset that has the most recent data pointing to HOME should do the failover. To change over all other filesets pointing to the same HOME, use the same command with the **-target-only** flag so they do not push data to HOME. The selected fileset copied data to HOME creating a new 1:1 copy of all the objects in the fileset. The failover command copies all of the data from the Cache to the HOME so the speed of this operation depends on the amount of data in the Cache and your network bandwidth. Data is transferred to HOME in the background.

Example 6-24 on page 307 shows a step-by-step procedure on how to fail over to a new HOME.

6.3.7 Temporary loss of Cache

In cases where an IW Cache is down temporarily due to disaster or maintenance, applications can be moved to access HOME. The nature of AFM is to push the data asynchronously, so the Cache might have pending updates that have not been pushed to HOME at the time the applications are moved to HOME. For this reason, if this is for a planned outage, you should issue a **flushpending** command before taking the Cache site down. This ensures that all outstanding changes are copied to HOME. Of course, if it is an unplanned outage, you do not have this option. For an unplanned outage, your application has to deal with the possibility that some changes have not yet been copied to HOME. In some situations, snapshots can help determine a consistent point in time for a data set. In

cases where application semantics allow them to move from Cache to HOME with pending updates still in Cache, it is possible that files at HOME will change along with the pending data in Cache.

In independent-writer mode, AFM periodically validates metadata information against HOME. When a revalidation is required, the requester has to wait for the validation to complete. On high latency networks, this can lead to slower than normal file opens.

The length of time between validations is configurable and can be tuned based on your application requirements. For example, if you operate your environment with AFM IW in an environment where you know that no data is changed in HOME, you can adjust the **afmFileLookupRefreshInterval** and the **afmDirLookupRefreshInterval** to large values so that content remains valid longer in Cache possibly reducing the frequency of revalidation messages.

One popular use for independent-writer mode is for disaster recovery (DR).

Figure 6-5 on page 324 illustrates a DR solution based on an AFM independent-writer fileset. In this scenario, the production workload is run on Side A (Cache) and the data is transferred automatically in the background by AFM to Side B (HOME). For planned maintenance, a switch over to HOME for the application load is relatively easy. First, stop the application on Side A and check that all entries in the AFM queues are transferred by issuing a **flushpending** command and monitoring it by using the **mmafmctl** command until you see the Queue Length is zero (0). Then, start the workload on HOME. By using fileset in independent-writer mode, changes in HOME are supported. So there are no more changes to AFM needed.

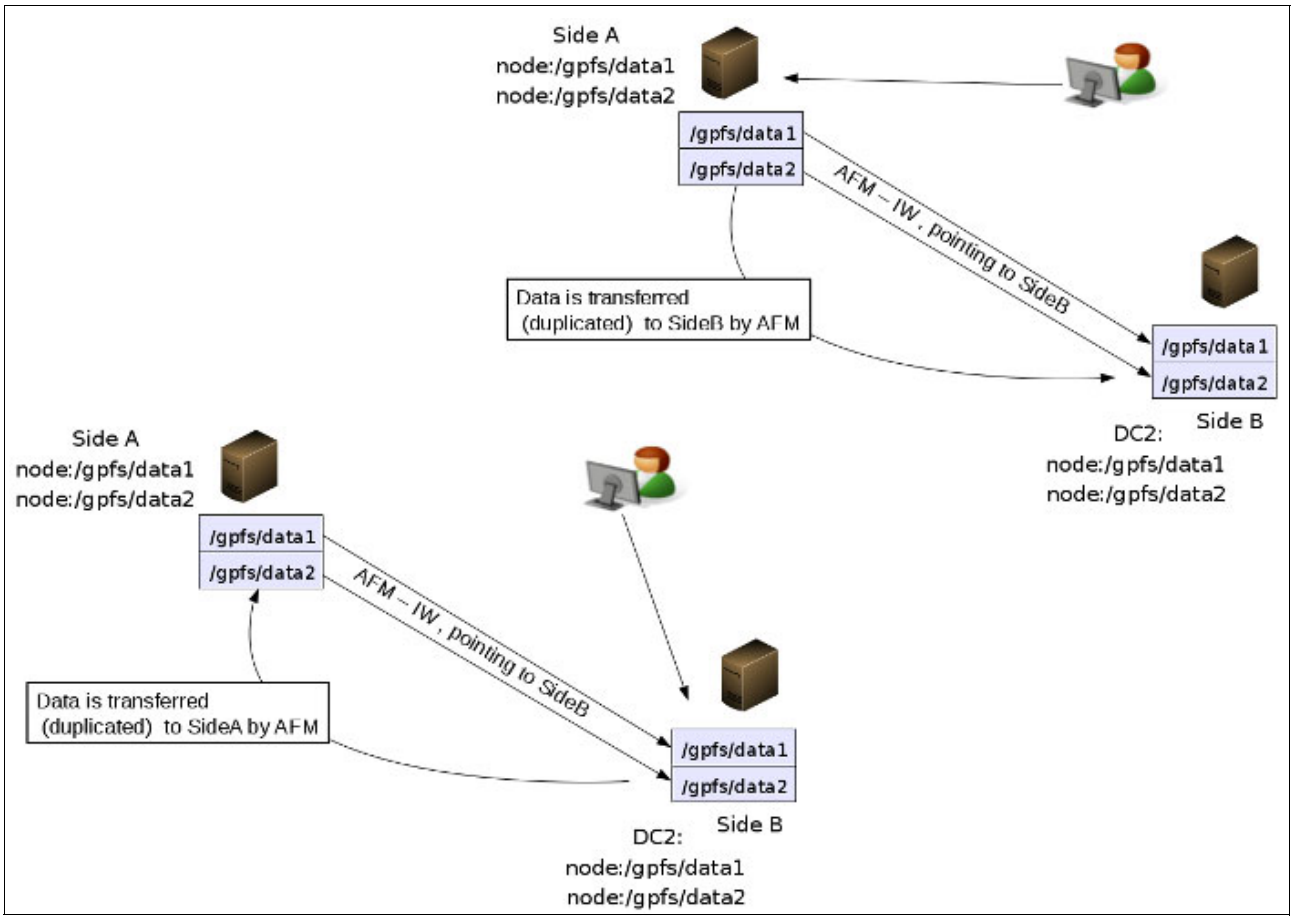


Figure 6-5 DR scenario, regular and DR workload

Once the original Side A is ready for use, you can move the production workload back to Side A. As the application is running files that are newer at HOME or exist at HOME, but not in Cache, these files are copied on demand. It is a good idea to issue a prefetch for all data that could have been created or changed during the outage in case the data is needed by the running application in a timely manner.

Note: A major advantage of AFM filesets in IW mode is that the filesets can be reused after changes in HOME because AFM keeps both sides synchronized.

6.3.8 Using AFM independent-writer for DR scenarios

Similar to the procedure described in 6.3.7, “Temporary loss of Cache” on page 322, you can use AFM’s capabilities for covering planned and unplanned downtimes of data centers.

An independent-writer fileset can be used to create a DR environment by using the Cache for the production workload and HOME as the DR site. If there is a failure of the Cache, the production workload is moved to use the data copy at HOME. If the HOME fails, the production workload continues to run in the Cache and outstanding changes are copied to HOME when it becomes available.

The following are the basics of independent-writer mode for DR:

- Cache is used for the read/write production workload.

- HOME is treated as read-only unless the Cache fails.
- Remote snapshots (**`mmpsnap`**) can be used to provide consistency points.
- To monitor RPO, you can time periodic **`flushpending`** commands or monitor the creation of remote snapshots at HOME.

The use of independent-writer for disaster recovery (DR) was briefly discussed in the last section. This section explores a DR scenario in more detail.

The nature of AFM is to push data asynchronously because the Cache might have pending updates that have not been pushed to HOME at the time the Cache fails. Some applications can handle moving to HOME with pending updates still in Cache.

When the Cache comes back, the first access causes the Cache to push stale data to HOME. This might result in an old update to overwrite the latest version of a file at HOME. In this case, the following steps should be followed so that the latest data is preserved and old data is discarded. AFM determines what is “old” data based on `mtime` file and failover time.

The failover time is provided by the administrator when running the **`mmafmctl failback`** command. The value provided represents the time when applications were moved to HOME.

This means that if AFM finds an update to a file in Cache that has a corresponding newer version at HOME, and that time in Cache is before the failover time, it throws out the Cache version.

For a better overview, we divide the DR scenario into three phases. In phase 1, the Cache fails as shown in Figure 6-6.

Phase I

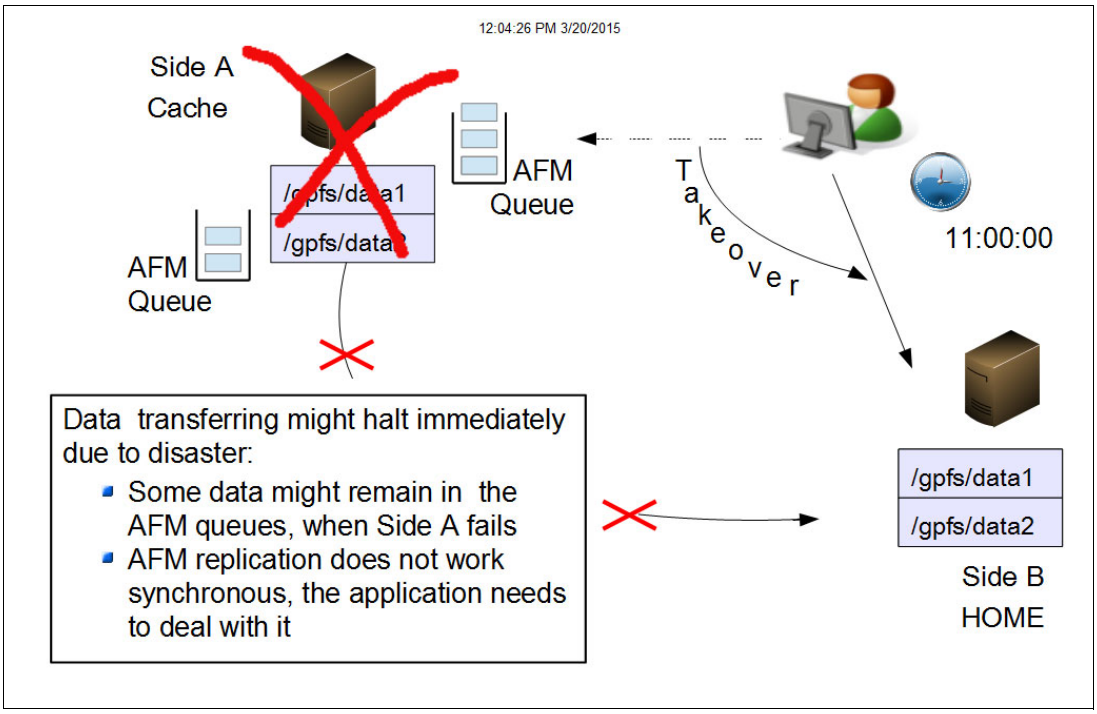


Figure 6-6 Phase one, Cache unexpectedly dies

Phase II

After the application is moved to HOME (done outside Spectrum Scale), it is important to note the time stamp when the application workload starts on HOME. In Figure 6-6 on page 325, the failover time is 11:00 a.m.

Note: It is important to track the date and time when HOME (Side B) is first accessed by the application in case of an outage on the Cache (Side A).

After the Cache side is back, synchronize the Cache fileset to ensure that no outdated data in the AFM queue is transferred to HOME overwriting data updates since the failure.

Phase III

This is the failback phase as follows:

1. When the Cache side is ready for failback, stop your application at HOME. From the GPFS/AFM perspective, the application workload can even continue in HOME during the failback of Cache and the application workloads can be switched back later.
2. Link the Cache site back to the HOME by bringing up the gateways. Do not access the fileset yet.
3. Run the **`mmafmctl failback`** command on the Cache to reconcile outstanding changes. Specify the time when the HOME became production including the complete time zone, in case HOME and Cache are in different time zones. See Figure 6-7.

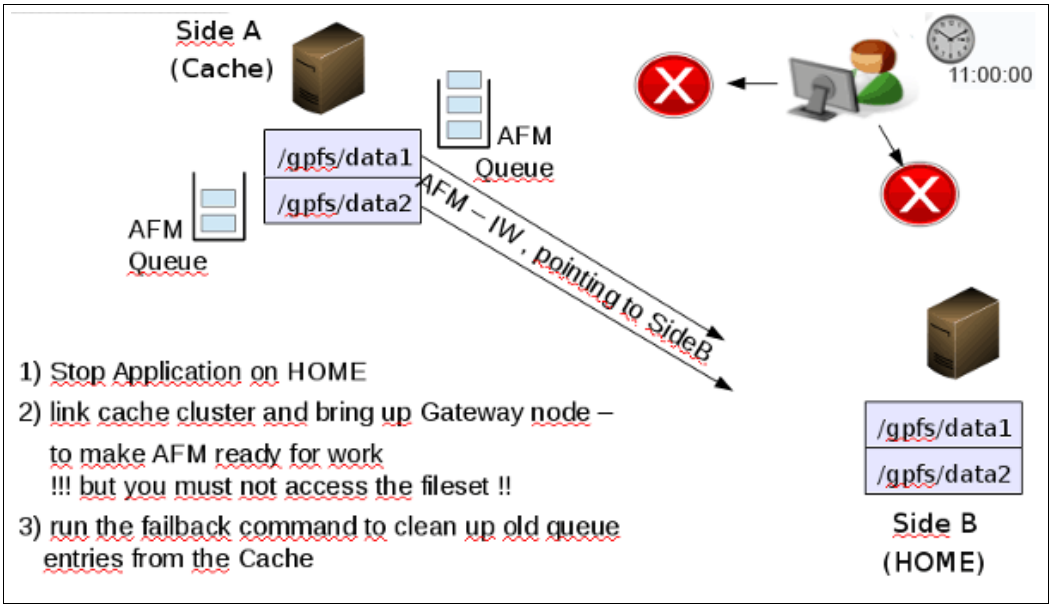


Figure 6-7 AFM DR failback scenario

Note: Do not access an independent-writer Cache fileset after a DR event without running failback.

The **`mmafmctl failback`** command resolves conflicts between pending updates in Cache based on the time of the failover with the data changes at HOME. The **`failback`** command only handles outstanding updates from the Cache throwing away old conflicts and flushing outstanding valid updates to HOME. Any files created or modified at HOME are fetched on the next access based on the revalidation interval. To maintain a full DR copy, prefetch the

new data from HOME into the Cache using the **mmafmctl prefetch** command. When the **failback** command is in progress, the Cache state changes several times as shown in Example 6-52 on page 328.

In the example, the DR scenario starts with creating a file on Cache, then verifying that AFM works properly (Example 6-50). Then, some data is added to the file and the Cache is failed hard stopping the Cache immediately. To ensure that there were outstanding changes, the test node crashed before the gateway node could transfer the data to HOME.

Example 6-50 Disaster recovery scenario (1/3)

```
On cache

node1:/gpfs/cache/fileset_IW # vi file_short_before_desaster  #<<<<<<<<< writing some stuff in a file
node1:/gpfs/cache/fileset_IW # cat file_short_before_desaster
last data from cache, short before crash
########################################

# verify, that AFM works and the data gets queued.. .. all works fine

node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Dirty           node2           4              46
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              51
#
#
# update the file and crash immediately (verify, that the Queue was filled)
node1:/gpfs/cache/fileset_IW # echo " last update on cache" >> file_short_before_desaster
node1:/gpfs/cache/fileset_IW # sleep 5 ; mmafmctl fs1 getstate; ssh node2 "init 0 &" ; init 0   # halt every node in the cluster immediately
```

Now that the Cache side is down, the focus turns to the HOME and the disaster recovery process. Example 6-51 shows the steps to reactivate the Cache side. During disaster recovery, you can work with HOME and change data.

Example 6-51 Disaster recovery scenario (2/3)

```
# on HOME :
node4:/gpfs/fshome/fset002new # ll
[...]
-rw-r--r-- 1 root root 83 Oct 23 06:33 file_short_before_desaster
node4:/gpfs/fshome/fset002new # cat file_short_before_desaster
last data from cache, short before crash<
########################################


assuming the application "rolls back and starts again, I start using this file on HOME"
# note down the time stamp
node4:/gpfs/fshome/fset002new # date
Thu Oct 23 06:35:20 CEST 2014

node4:/gpfs/fshome/fset002new # echo " written on HOME during DR" >> file_short_before_desaster
node4:/gpfs/fshome/fset002new # cat file_short_before_desaster
text and data
####################
```

```
 written on HOME during DR

node4:/gpfs/fshome/fset002new #
node4:/gpfs/fshome/fset002new # ls -l
[...]
-rw-r--r-- 1 root root 191 Oct 23 06:36 file_short_before_desast
node4:/gpfs/fshome/fset002new #
```

When the Cache side is repaired, start Spectrum Scale including the gateway nodes as shown in Example 6-52. Verify that the connectivity to HOME is available by manually mounting the NFS share or accessing the remotely mounted file system. Again, do not access the fileset, for example do not **cd** to the directory to see if it is working, wait. The file system has to be mounted and the fileset linked.

Concerning your needs, you might stop the application on HOME to be prepared to be able to switch back the application immediately after the failback. It is also possible to switch back the application later because changes on HOME with independent-writer are supported. Proceed as follows.

Example 6-52 Disaster recovery scenario (3/3)

```
node1:~ # mmstartup -a
Thu Oct 23 06:40:08 CEST 2014: mmstartup: Starting GPFS ...
node1:~ # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      node1            active
       2      node2            active

node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Inactive

node1:~ # mmafmctl fs1 failback -j fileset_IW --start --failover-time "Thu Oct 23 06:35:20 CEST 2014"

node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec

------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      FailbackInProg  node2           0              0
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      FailbackInProg  node2           0              0
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      FailbackCompleted node2         0              1
node1:~ #
```

If the fileset does not reach the “FailbackCompleted” state, check the log files for errors (`/var/adm/ras/mmfs.log*`). The failback process can take a while, depending on the number of changes and network performance. You can get additional information from the gateway nodes’ mmfs.log for progress information and to see what task is running (see Example 6-53 on page 329).

Example 6-53 Successful recovery entries in mmfs.log

```
Thu Oct 23 06:43:17 CEST 2014: [N] AFM: Starting recovery for fileset 'fileset_IW' in fs 'fs1'
Thu Oct 23 06:43:17 CEST 2014: mmcommon afmrecovery invoked: device=fs1 filesetId=1 filesetName=fileset_IW
resync=0 recoverySeq=1031084150 snap=1 mode=6 fsType=0 drpsnap=
Thu Oct 23 06:43:17 CEST 2014: [N] AFM: Running following policy scan to find missing updates from
/gpfs/cache/fileset_IW
Thu Oct 23 06:43:17 CEST 2014: [N] AFM: mmafmlocal /usr/lpp/mmfs/bin/mmapplypolicy
/gpfs/cache/fileset_IW/.snapshots/fileset_IW.afm.4279 --scope fileset -S fileset_IW.afm.4279 -L 0 -I defer
-f /var/mmfs/afm/fs1-1/recovery/policylist.data -N mount -g /gpfs/cache/fileset_IW/.ptrash --iscan-by-number
-P /var/mmfs/afm/fs1-1/recovery/recoveryPolicy >/var/mmfs/afm/fs1-1/recovery/policyLog 2>&1
Thu Oct 23 06:43:31 CEST 2014: [I] AFM: /usr/lpp/mmfs/bin/tspcachescan fs1 fileset_IW 1 0 1031084150
fileset_IW.afm.4279 6 0
Thu Oct 23 06:43:31.269 2014: [I] AFM: Detecting operations to be recovered...
Thu Oct 23 06:43:31.274 2014: [I] AFM: Found 1 update operations...
Thu Oct 23 06:43:31.299 2014: [I] AFM: Starting 'queue' operation for fileset 'fileset_IW' in filesystem
'fs1'.
Thu Oct 23 06:43:31.300 2014: [I] Command: tspcache fs1 1 fileset_IW 0 3 1031084150 1 0 23
Thu Oct 23 06:43:31.343 2014: [I] Command: successful tspcache fs1 1 fileset_IW 0 3 1031084150 1 0 23
Thu Oct 23 06:43:31 CEST 2014: [I] AFM: Finished queuing recovery operations for /gpfs/cache/fileset_IW
```

Verify that no errors are written to the mmfslog. To enable the workload again on Cache, set back the queue into the active state with the **mmafmctl** command. See Example 6-54.

Example 6-54 Set queue state back to Active

```
node1:~ # mmafmctl fs1 failback -j fileset_IW --stop
node1:~ # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://node4/gpfs/fshome/fset002new      Active          node2           0              1
node1:~ #
```

From here on, you are back to regular state and the workload can continue working in Cache. However, during the failback, no access is possible on the Cache side. The last step as shown in Example 6-55 is to verify that all updates to the file done in HOME during the outage of Cache are synced accordingly, and are available again in Cache as well.

Example 6-55 Verify the files content

```
node2:/gpfs/cache/fileset_IW # cat file_short_before_desaster
last data from cache, short before crash
########################################
 written on HOME during DR
node2:/gpfs/cache/fileset_IW #
```

Some additional hints are presented as follows.

Each AFM fileset is independently managed and has a one-to-one relationship with a target (HOME). This allows different protocol backends to coexist on separate filesets in the same file system. However, AFM does not validate the target for correctness when a fileset is created. It is the responsibility of the user to specify a valid target. Using a Spectrum Scale target that belongs to the same file system as the AFM fileset might lead to undesirable consequences.

6.3.9 Summary

Independent-writer filesets provide the flexibility of building a global name space (see 6.6, “Building a global name space” on page 337). By having the change of a supported resynchronization with HOME, you can use AFM with independent-writer to build disaster recovery capable environments.

6.4 Using AFM to migrate the content of an existing file system

AFM migration allows you to migrate data from any earlier storage appliance to a Spectrum Scale cluster using the NFS protocol. This can be useful if the storage appliance is old or going off lease, and needs to be replaced. To make the migration as seamless as possible, when the migration is complete, you can move the IP addresses of the appliance to Spectrum Scale so users and applications are the least affected. AFM NFS migration can be useful when migrating data from an existing file system or NAS appliances into a Spectrum Scale cluster or to move data to a new Spectrum Scale file system with new settings, such as blocksize, inode size, or any other attribute, which cannot be changed dynamically on an existing file system.

AFM provides an option to do a planned migration with a minimal downtime by pulling data on demand by prefetching data. In the case of dynamic migration, the data is moved slowly based on access over time. Once the data is moved, the source of migration can be removed. See Figure 6-8.

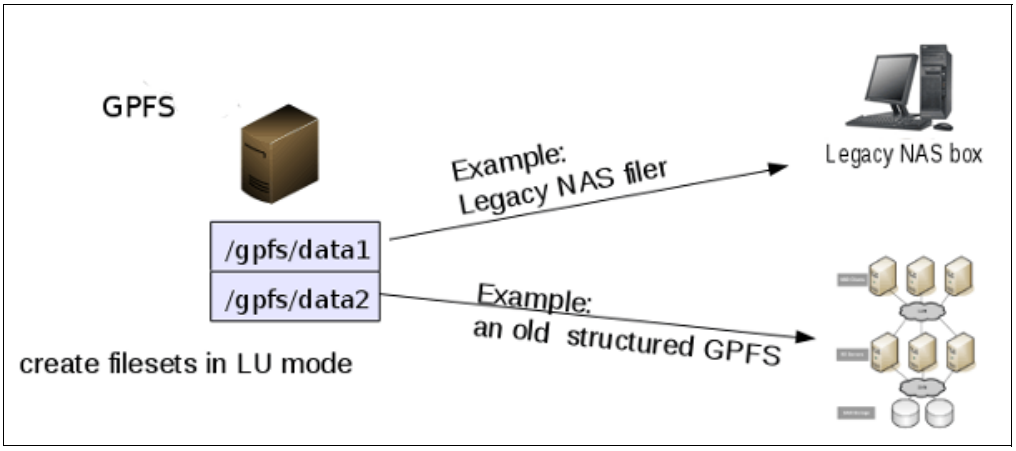


Figure 6-8 Various migration scenarios by AFM

On a Spectrum Scale data source, AFM moves all Spectrum Scale extended attributes and ACLs, and file sparseness is maintained. AFM does not migrate file system-specific parameters including quotas, snapshots, file system level tuning parameters, policies, fileset definitions, encryption keys, or dmapi parameters. To maintain hard links, use the **prefetch** command with the **--ns** option (see **mmafmctl prefetch** command in the Spectrum Scale documentation).

AFM migrates data as root so it can set the proper permission on the copied file. Pre-allocated files at HOME are not maintained; that is, only actual data blocks are migrated based on file size. User and group definitions need to be the same at the source and target since these file attributes are included when a file is copied.

On a non-Spectrum Scale data source, POSIX permissions or ACLs are pulled. AFM does not migrate NFS V4/SMB ACLs, non-posix file system attributes, and sparseness.

In general, three steps are needed to migrate data summarized as follows:

- Incremental
- Progressive migration
- Progressive migration with (almost) no downtime

The following steps outline the third migration method: A progressive migration with no downtime:

1. Create an AFM local-update (LU) fileset with the AFM target pointing to an NFS v3 data source.
2. Move applications from the data source to the AFM Cache fileset or optionally prefetch critical file data by using the **mmafmctl --prefetch** command. Then, move applications.
3. Data is migrated on demand. At the same time, the administrator can migrate critical data from the source by using the **mmafmctl --prefetch**.

A short downtime is required twice: first, to move applications to the new cluster, and second, once all the data is migrated to turn off AFM. Until all of the data has been fetched into the new fileset, applications can experience higher latency when opening a file since it is automatically pulled from HOME. The key here is to make sure that HOME stays connected until a complete copy of the data is present in the Cache to avoid application errors when accessing uncached files and to avoid any namespace conflicts due to disconnection.

The UIDs should be maintained across the source and target cluster. This means if a file is owned by user ID (UID) 10, the same UID owns it in Cache. Therefore, it is important that both systems have same UID-to-user mapping. In addition to UIDs, other file attributes are maintained when data is copied into a Cache or pushed to HOME. For example, the mtime of a file is maintained from source to target.

Example 6-56 shows how AFM can be used to migrate data into a new Spectrum Scale by using an LU AFM fileset. The example shows the exchange of an old NAS appliance with a new, more powerful clustered environment. The content of that earlier NAS appliance is accessible by NFS from our new environment so we create a new Spectrum Scale file system with a new fileset. You should be familiar with AFM basics and managing filesets. If not, review this chapter from the beginning or at least from section 6.2.1, “Creating a single-writer Cache fileset” on page 299 and also the following paragraphs.

The example starts by creating the local-update fileset Cache. Example 6-56 includes the details about creating and linking the new fileset. For this example, node4 is the earlier NAS appliance and node2 belongs to the Spectrum Scale cluster containing the new fileset.

Example 6-56 Create a fileset for migration purposes (LU)

```
node2:/gpfs/cache # mmcrfileset fs1 NewHighPerf -p afmmode=lu -p afmtarget=node4:/old_poor_performing_NAS --inode-space  new
Fileset NewHighPerf created with id 1 root inode 131075.
node2:/gpfs/cache # mmlsfileset fs1
Filesets in file system 'fs1':
Name                    Status    Path
root                    Linked    /gpfs/cache
NewHighPerf             Unlinked  --
node2:/gpfs/cache # mmlsfileset fs1 NewHighPerf
Filesets in file system 'fs1':
Name                    Status    Path
NewHighPerf             Unlinked  --
```

```
node2:/gpfs/cache # mmlinkfileset fs1 NewHighPerf
Fileset NewHighPerf linked at /gpfs/cache/NewHighPerf
node2:/gpfs/cache # mmlsfileset fs1 NewHighPerf
Filesets in file system 'fs1':
Name                    Status    Path
NewHighPerf             Linked    /gpfs/cache/NewHighPerf
```

Once the fileset is linked, it is ready for use and the application load can be run by using the data in Spectrum Scale. When a non-cached file is opened by AFM automatically, there might be additional latency the first time the file is opened, but file operations should run as normal.

Check your fileset attributes and state as shown in Example 6-57.

Example 6-57 File attributes in AFM

```
node2:/gpfs/cache # mmlsfileset fs1 NewHighPerf --afm -L
Filesets in file system 'fs1':

Attributes for fileset NewHighPerf:
===================================
Status                                  Linked
Path                                    /gpfs/cache/NewHighPerf
Id                                      1
Root inode                              131075
Parent Id                               0
Created                                 Thu Oct 23 17:26:21 2014
Comment
Inode space                             1
Maximum number of inodes                93184
Allocated inodes                        66560
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  nfs://node4/old_poor_performing_NAS
Mode                                    local-updates
File Lookup Refresh Interval            30 (default)
File Open Refresh Interval              30 (default)
Dir Lookup Refresh Interval             60 (default)
Dir Open Refresh Interval               60 (default)
Async Delay                             disable
Last pSnapId                            0
Display Home Snapshots                  no (default)
Number of Gateway Flush Threads         4
Prefetch Threshold                      0 (default)
Eviction Enabled                        yes (default)
node2:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS     Inactive
```

Now that the fileset is linked, activate it by simply entering the directory. To demonstrate how this works, first the content on the NAS appliance (node4) is displayed by listing the directory. Then, the associated directory is listed in the Cache and all of the same directory contents are available immediately. At this point, the inodes information has been copied into the Cache but not the file data. Now you can start creating new files in the Cache. No conflicts occur because the metadata information has already been copied as shown in Example 6-58 on page 333.

Example 6-58 AFM fetch data process

```
## content of old legacy NFS
node4:/old_poor_performing_NAS # ll
total 12
-rw-r--r-- 1 root root 30 Oct 23 17:16 file1
-rw-r--r-- 1 root root 30 Oct 23 17:16 file2
-rw-r--r-- 1 root root 30 Oct 23 17:16 file3
node4:/old_poor_performing_NAS #


## on Cache , new side:
##now activating the fileset by entering the directory


node2:/gpfs/cache # cd NewHighPerf/
node2:/gpfs/cache/NewHighPerf # ll
total 65
drwx------ 65535 root root 32768 Oct 23 17:31 .afm
-rw-r--r--     1 root root    30 Oct 23 17:16 file1
-rw-r--r--     1 root root    30 Oct 23 17:16 file2
-rw-r--r--     1 root root    30 Oct 23 17:16 file3
drwx------ 65535 root root 32768 Oct 23 17:31 .ptrash
dr-xr-xr-x     2 root root 32768 Jan  1  1970 .snapshots
node2:/gpfs/cache/NewHighPerf #

node2:/gpfs/cache/NewHighPerf # mmafmctl fs1 getstate
Fileset Name    Fileset Target                            Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                            -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS       Active          node2           0              6
node2:/gpfs/cache/NewHighPerf #


# start writing additionally files to the new fileset
node2:/gpfs/cache/NewHighPerf # echo "dkfkdjf3344" > filenew_fastPerf
node2:/gpfs/cache/NewHighPerf # ll
total 65
drwx------ 65535 root root 32768 Oct 23 17:31 .afm
-rw-r--r--     1 root root    30 Oct 23 17:16 file1
-rw-r--r--     1 root root    30 Oct 23 17:16 file2
-rw-r--r--     1 root root    30 Oct 23 17:16 file3
-rw-r--r--     1 root root    12 Oct 23 17:32 filenew_fastPerf
drwx------ 65535 root root 32768 Oct 23 17:31 .ptrash
dr-xr-xr-x     2 root root 32768 Jan  1  1970 .snapshots
```

File data is copied from HOME when requested. The request can be simply reading the file or you can use the **mmfsadm prefetch** command. In the example (just three files), the **cat** command is used to read the files.

In an AFM local-update, fileset changes on Cache are not copied to HOME. All updates are kept locally, this is why local-update mode is used for migrations. The data is copied from the legacy device into Spectrum Sale, and there is no need to copy any new data back to the NAS appliance. In Example 6-59 on page 334, the **mmfsadm getstate** command is used to show that the Queue Length remains 0, regardless how often data is created or changed in the Cache. If a file is modified in the Cache for which the file data has not already been copied, the data is first copied from HOME before the update can occur.

Example 6-59 AFM behavior in local-update mode

```
node2:/gpfs/cache/NewHighPerf # echo "dkfkdjf3344" > filenew_fastPerf
node2:/gpfs/cache/NewHighPerf # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS     Active          node2           0              6
node2:/gpfs/cache/NewHighPerf # echo "dkfkdjf3344" > filenew_fastPerf
node2:/gpfs/cache/NewHighPerf # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS     Active          node2           0              6

# fetch in the data of the files
node2:/gpfs/cache/NewHighPerf # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS     Active          node2           0              6
node2:/gpfs/cache/NewHighPerf # cat file1 file2 file3
Thu Oct 23 17:16:35 CEST 2014
Thu Oct 23 17:16:36 CEST 2014
Thu Oct 23 17:16:36 CEST 2014
node2:/gpfs/cache/NewHighPerf # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
NewHighPerf     nfs://node4/old_poor_performing_NAS     Active          node2           0              12
```

In the second part of the example, you see from the Queue numExec column how you can force to read in the data.

Usually, you would use the Spectrum Scale policy engine for identifying every object in the source file system. With this list, you can instruct AFM to prefetch data in the background. When HOME is a non GPFS environment, you can generate the file list with the **find** command or any other tool. So you can migrate millions of files in the background. Regardless of how long the migration takes, it is guaranteed that if an application accesses a file, the content is always the last recent one. Either already prefetched from HOME or read in on access in that moment. After assuring that the data is completely migrated, turn off AFM for that fileset and recycle the old environment.

The final step of data migration is to disable AFM on the fileset. When you confirm that all of the data has been migrated, disable AFM on the fileset before completely retiring the NAS appliance. Once an AFM fileset is disabled, it cannot be reenabled. To disable AFM on a fileset, the fileset needs to be unlinked (see Example 6-60), which requires a short downtime because the fileset is not accessible when it is unlinked.

Example 6-60 Turning off AFM

```
node2:~ # mmunlinkfileset fs1 NewHighPerf
Fileset NewHighPerf unlinked.
node2:~ # mmchfileset fs1 NewHighPerf -p afmTarget=disable
Fileset NewHighPerf changed.
node2:~ # mmlinkfileset fs1 NewHighPerf -J /gpfs/cache/NewHighPerf
Fileset NewHighPerf linked at /gpfs/cache/NewHighPerf
node2:~ # mmlsfileset fs1 NewHighPerf --afm -L
Filesets in file system 'fs1':
```

```
Attributes for fileset NewHighPerf:
===================================
Status                                  Linked
Path                                    /gpfs/cache/NewHighPerf
Id                                      1
Root inode                              131075
Parent Id                               0
Created                                 Thu Oct 23 17:26:21 2014
Comment
Inode space                             1
Maximum number of inodes                93184
Allocated inodes                        66560
Permission change flag                  chmodAndSetacl
afm-associated                          No
node2:~ #
```

The fileset is a regular independent fileset. The preceding method is an efficient way of migrating data from an outdated NAS appliance into IBM Spectrum Scale.

6.5 Customizing and tuning AFM filesets

There are AFM tuning parameters that are globally set by using the **mmchconfig** command, and others specific to each fileset set by using the **mmchfileset** command. Where there is a fileset parameter that matches a global parameter, the global parameter is treated as default, and a parameter set on an individual fileset overrides the global default. A fileset-only parameter is the number of inodes. As an AFM fileset must have its own inode space, it can be useful to verify inode utilization in the fileset as shown in Example 6-61.

Example 6-61 Detailed listing of fileset attributes

```
node1:/gpfs/cache/fileset_SW # mmlsfileset fs1 fileset_SW
Filesets in file system 'fs1':
Name                    Status    Path
fileset_SW              Linked    /gpfs/cache/fileset_SW
node1:/gpfs/cache/fileset_SW # mmlsfileset fs1 fileset_SW -L
Filesets in file system 'fs1':
Name                         Id      RootInode  ParentId Created                     InodeSpace      MaxInodes    AllocInodes Comment
fileset_SW                    1         131075         0 Thu Oct  9 20:14:06 2014       1               70144          66560
```

To check AFM-specific details about a fileset, use the **-afm** flag with the **mmlsfileset** command (Example 6-62). You can change these parameters to customize the behavior of the AFM fileset. A useful set of configuration parameters to explore are the refresh intervals, though for most installations, the default values works fine.

Example 6-62 Listing the AFM-related values with the mmlsfileset command

```
node1:/gpfs/cache/fileset_SW # mmlsfileset fs1 fileset_SW --afm
Filesets in file system 'fs1':
Name                    Status    Path                                    afmTarget
fileset_SW              Linked    /gpfs/cache/fileset_SW                  nfs://node4/gpfs/fshome/fset001
node1:/gpfs/cache/fileset_SW # mmlsfileset fs1 fileset_SW --afm -L
Filesets in file system 'fs1':

Attributes for fileset fileset_SW:
```

```
==================================
Status                                  Linked
Path                                    /gpfs/cache/fileset_SW
Id                                      1
Root inode                              131075
Parent Id                               0
Created                                 Thu Oct  9 20:14:06 2014
Comment
Inode space                             1
Maximum number of inodes                70144
Allocated inodes                        66560
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  nfs://node4/gpfs/fshome/fset001
Mode                                    single-writer
File Lookup Refresh Interval            30 (default)
File Open Refresh Interval              30 (default)
Dir Lookup Refresh Interval             60 (default)
Dir Open Refresh Interval               60 (default)
Async Delay                             15 (default)
Last pSnapId                            0
Display Home Snapshots                  no
Number of Gateway Flush Threads         4
Prefetch Threshold                      0 (default)
Eviction Enabled                        yes (default)
node1:/gpfs/cache/fileset_SW #
```

The refresh parameters affect how often AFM checks data consistency with HOME when using read-only, local-update, and independent-writer mode filesets. On a high latency network, if your workload supports it, increase the revalidation intervals. Alternatively, if the application requires tighter consistency, lower these values. Keep in mind that the lower the value, the more often network messages are sent to HOME, which can affect performance. Since single-writer does not perform revalidation, these filesets are unaffected by these parameters. Table 6-3 provides an overview on changeable parameter.

Table 6-3 Tunables

Setting	[f]ileset - [s]ystem - default	Short description
afmTarget	[f] na	afmTarget can only be changed in special situations with **mmchfs**/**mmchfileset** (such as failover of a HOME).
afmState	[f] na	This indicates whether the fileset is enabled for pcache. By default, afmState is enabled. It is disabled in special situations (such as file system restore from a snapshot) to ensure that Cache does not get updated when the relationship with HOME is compromised. afmState can also be explicitly disabled for testing purposes or other conditions where a sync to HOME is to be disallowed.
afmmode	[s]	This defines the mode in which the Cache operates (see Panache Modes of Operation for details). For conflict handling, Cache is considered to be in conflict if data is independently modified on the Cache and HOME cluster since they were synced last.

Setting	[f]ileset - [s]ystem - default	Short description
afmFileLookupRefreshInterval, afmDirLookupRefreshInterval	[s]	Data revalidation takes places when an object is looked up through **ls**, **stat**, or other similar operations. Refresh Interval controls how frequent revalidation should happen on lookup, especially for the situation where HOME cluster data is not changing frequently. For the scenarios where HOME cluster data changes frequently, refresh intervals might be set to 0.
afmFileOpenRefreshInterval, afmDirOpenRefreshInterval	[s]	Similar to Lookup Refresh Interval, Open Refresh Interval is used to revalidate data when an object is opened for I/O operations, such as reads and writes. For an already Cached file, it determines how frequently OPEN requests must go to a gateway node instead of directly reading data from Cache. Setting a lower value for Open Refresh Interval guarantees a higher level of consistency.
afmExpirationTimeout	[s]	These options control how long a network outage between the Cache and HOME can be tolerated before the data in the Cache is considered out of sync with HOME. The Cache waits until afmDisconnectTimeout expires before declaring a network outage. At this point, only cached data can be accessed, reading uncached data results in I/O errors, and new updates remain in the queue waiting for a reconnect. Cached data is only available until afmExpirationTimeout expires, at which point the cached data is considered “expired” and cannot be read until a reconnect occurs. Note, not applicable for single-writer.
afmDisconnectTimeout	[s] 60	It is set by **mmchconfig** and controls the time AFM tolerates until the connection state is considered as disconnected. It might be useful to lower to 30 seconds.
afmAsyncDelay		Operations such as writes are asynchronous regarding remote cluster. The time for which they can be delayed is specified by using afmAsyncDelay. This delay can be useful for write-intensive applications that keep writing to same set of files. Delaying writes to HOME cluster gives us an opportunity to send a single-write containing the latest data while rejecting the older ones, if the writes are done to the same part of the file. On the downside, setting a very high value weakens the consistency of data on remote cluster.
afmParallelReadThreshold	[s] - 1 GB	Panache supports parallel reads from multiple gateway nodes for faster caching of files. This value sets the file size above which parallel reads are used. A value of “disable” disables parallel reads.
AfmNumFlushThreads	[s] - 4	You can increase Flush Threads. Therefore, you have many threads to handle your changes to transfer.

The **afmFileLookupRefreshInterval**, **afmDirLookupRefreshInterval**, **afmFileOpenRefreshInterval**, **afmDirLookupRefreshInterval**, **afmExpirationTimeout**, **afmAsyncDelay**, and **afmParallelReadThreshold** parameters are all system-wide (set by using the **mmchconfig** command) and fileset-specific (set by using the **mmchfs**/**mmchfileset** command). If a fileset has some fileset-specific parameters set, they take precedence over their system-wide counterparts. The **afmDisconnectTimeout** parameter is system-wide only.

6.6 Building a global name space

The basic AFM elements (HOME and Cache) can be combined to create a global namespace. Global namespace in this context means that any client node in the world can use the same path to connect to the data within any of the IBM Spectrum Scale clusters that are part of the name space as shown in Figure 6-5 on page 324.

You can connect clusters by using NFS or Spectrum Scale multi-cluster. The benefit of using AFM is improved performance even with high latency or unreliable networks, and the ability to continue to access cached data when the WAN connection is down. AFM can improve application performance at a remote site using the standard Cache behaviors. In addition, depending on the AFM mode, you can build up new DR scenarios by simply shipping the data to a single place where all the backup or other post processing with the data can be done. Refer to Figure 6-9.

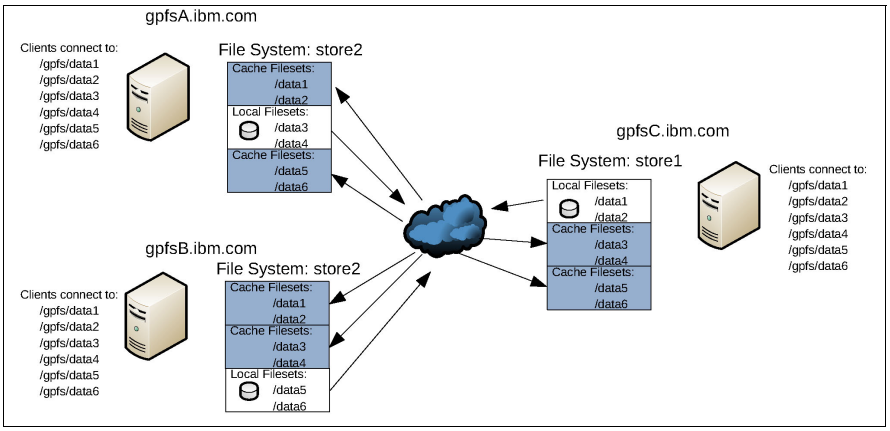


Figure 6-9 Sharing file systems between clusters

6.7 Enhancements in IBM Spectrum Scale 4.1 TL 1

With IBM Spectrum Scale 4.1.0.4 (Released in Q4 2014), some great enhancements were introduced to Spectrum Scale in terms of *performance* and *availability*.

6.7.1 Gateway node changes

In older GPFS releases (as of 3.5), the gateway (GW) node joins/leaves results in charge of Metadata Server (MDS) for the filesets. The change of MDS drives new MDS to run AFM fileset recovery as the old MDS would have dropped queued messages. So you hit into this issue when taking down one of the gateways, which means your entire fileset was running the recovery.

The fileset is still usable during a gateway failover; however, there some performance impacts during the failover runs.

With IBM Spectrum Scale 4.1, an enhancement was introduced to AFM. The enhancement transfers the memory queue to the new MDS thus avoiding the need for expensive AFM fileset recovery. This means from TL1 onwards, you can easily shut down one of your gateway nodes for maintenance without triggering the recovery because the queue is transferred gracefully to other gateway nodes.

As already stated, it is advised to have at least two gateway nodes in the cluster. Keep in mind that the configuration in HOME has to reflect all the gateway nodes in terms of NFS exports, permissions, and network connectivity.

You should have already added further gateway nodes in HOME with the **`mmchnode --gateway`** command as shown in Example 6-63 on page 339.

Example 6-63 Adding gateway nodes

```
node2:/gpfs/cache/NewHighPerf # mmlscluster
[...] some output depressed[...]

 Node  Daemon node name  IP address  Admin node name  Designation
-----------------------------------------------------------------
   1   node1.site        10.0.0.111  node1.site       quorum-manager
   2   node2.site        10.0.0.112  node2.site       gateway

node2:/gpfs/cache/NewHighPerf # mmchnode --gateway -N node1
Thu Oct 23 22:42:12 CEST 2014: mmchnode: Processing node node1.site
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
node2:/gpfs/cache/NewHighPerf # mmlscluster
[...] some output depressed[...]

 Node  Daemon node name  IP address  Admin node name  Designation
-----------------------------------------------------------------
   1   node1.site        10.0.0.111  node1.site       quorum-manager-gateway
   2   node2.site        10.0.0.112  node2.site       gateway

node2:/gpfs/cache/NewHighPerf #
```

Once a gateway is defined, there is no need for further configuration. There are some new parameters in version 4.1.0.4 that are not enabled by default. There is a new version of the hashing algorithm that defines how filesets are spread across gateways. When a cluster is upgraded, the old version one algorithm stays in effect. To move to the improved algorithm, use the **mmchconfig** command and set **afmHashVersion=2**. Only change this parameter if all of the nodes and clusters accessing this file system are at Spectrum Scale version 4.1.0.4 or newer. For the new hash version to take effect, the fileset needs to be relinked or the file system remounted.

6.7.2 Native GPFS protocol

GPFS 3.5 TL3 and earlier uses NFSv3 as the communication protocol between Cache and HOME. Using NFS has its advantages because it is a well-established standard protocol, and it allows HOME to be any NFSv3 export. However, NFSv3 lacks support for some file attributes (such as ACLs, xattrs, file sparseness). In order to support these, AFM uses a separate channel (called the “ctl” interface) to communicate with HOME. Going forward, as additional GPFS specific file systems and file metadata (such as filesets, quotas, snapshots, clones) need to be replicated, use of NFS as the transport mechanism has serious shortcomings. Even with NFSv4, which provides features such as ACLs and delegations, the need for a separate security infrastructure (such as Kerberos) can be prohibitive. Moreover, performance over the WAN in the face of higher network latencies proves to be a challenge, and requires careful setting of control and tuning parameters, both for NFS and TCP.

In Spectrum Scale 4.1, you can configure AFM to run with the native Network Shared Disk (NSD) protocol. This is implemented by mounting the remote HOME file system using multi-cluster, but the remote mounted file system is accessed by the gateway functions only.

To use the NSD protocol with AFM, you need to configure a remote mount or multi-cluster environment (refer to the GPFS documentation “Mounting a file system owned and served by another GPFS cluster” at the following web page: http://ibm.co/1FMs7I4). Once the remote file system is mounted on the gateway nodes, create a fileset using the NSD protocol (see Example 6-64 on page 340). Start by verifying that a valid multi-cluster configuration is available.

As opposed to the NFS-based connectivity, when using GPFS protocol, you have to assure that:

- Remote cluster NSDs need an accessible NSD server configuration.
- All NSDs must be a reachable over the network or be direct attached.

The **mmremotefs** command is used to verify that the remote file system is accessible as shown in Example 6-64.

Example 6-64 Verify the multi-cluster configuration

```
node1:/var/mmfs/ssl # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      node1            active
       2      node2            active
node1:/var/mmfs/ssl # mmremotefs show all
Local Name  Remote Name  Cluster name       Mount Point        Mount Options    Automount  Drive  Priority
remoteHOME  fshome       node3.home         /remote/fshome     rw               no           -        0

node1:/var/mmfs/ssl # mmmount remoteHOME -a
Thu Oct 23 23:28:32 CEST 2014: mmmount: Mounting file systems ...


node1:/var/mmfs/ssl # df -k
[...]
/dev/fs1          5726208  563200   5163008  10% /gpfs/cache
/dev/remoteHOME   5242880  482304   4760576  10% /remote/fshome
node1:/var/mmfs/ssl #
```

In Example 6-64, a valid multi-cluster configuration is verified. Next, for demonstration the remote file system is unmounted before creating the new fileset (see Example 6-65) by using the NSD (GPFS) protocol as the transport layer.

Example 6-65 Create AFM with GPFS protocol as the transport layer

```
node1:/var/mmfs/ssl # mmcrfileset fs1 fileset_IW -p afmtarget=gpfs:///remote/fshome -p afmmode=iw --inode-space=newFileset fileset_IW created
with id 1 root inode 131075.
node1:/var/mmfs/ssl # cd /gpfs/cache
node1:/gpfs/cache # mmlinkfileset fs1 fileset_IW
Fileset fileset_IW linked at /gpfs/cache/fileset_IW
node1:/gpfs/cache # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      gpfs:///remote/fshome                   Inactive
node1:/gpfs/cache # cd fileset_IW
node1:/gpfs/cache/fileset_IW # ll
total 97
drwx------ 65535 root root 32768 Oct 23 23:47 .afm
drwx------ 65535 root root 32768 Oct 23 23:47 .pconflicts
drwx------ 65535 root root 32768 Oct 23 23:47 .ptrash

dr-xr-xr-x     2 root root 32768 Jan  1  1970 .snapshots
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      gpfs:///remote/fshome                   Unmounted       node1           0              0
node1:/gpfs/cache/fileset_IW #
```

When running AFM with the NSD protocol, the administrator is responsible for the multi-cluster configuration and mounting of HOME file systems. AFM never tries to mount the remote IBM Spectrum Scale file system. Example 6-66 demonstrates how the fileset becomes *Active* when the file system is mounted on the gateway node.

Example 6-66 mmremote cluster mount

```
node1:/gpfs/cache/fileset_IW #mmmount remoteHOME -a
# access the fileset / wait 60 seconds
node1:/gpfs/cache/fileset_IW # mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      gpfs:///remote/fshome                   Active          node1           0              13
node1:/gpfs/cache/fileset_IW #
```

6.7.3 Performing parallel data transfers

To improve data transfer performance using AFM, you can now take advantage of the parallel I/O feature of IBM Spectrum Scale 4.1.0.4. To get an idea of how parallel I/O works in AFM, refer to Figure 6-10.

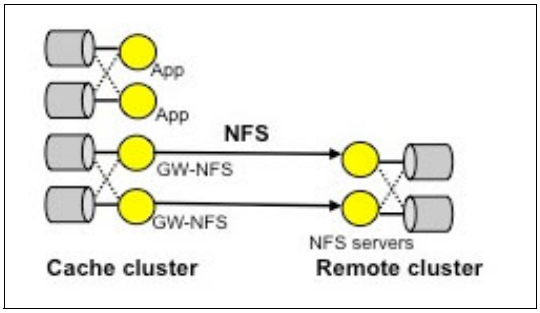


Figure 6-10 Parallel scale I/O

Parallel reads

Multiple gateway nodes can be used to prefetch a single file. This can improve performance when essentially transferring data between a single Cache node and HOME as the NFS server cannot saturate all available network and storage bandwidth.

In essence, *all* of the following statements must be true before parallel reads are enabled:

- The network bandwidth between the Cache and HOME edge routers is greater than the bandwidth between the GW and the Cache edge router.
- Home storage aggregate read bandwidth is greater than the end-to-end network bandwidth between a single GW and HOME NFS server.
- Cache aggregate write bandwidth is larger than the end-to-end network bandwidth between a single GW and HOME NFS server.

Parallel writes

Multiple gateway nodes can be used to synchronize changes to small and large files. This improves performance when transferring data between a single Cache node and HOME as the NFS server cannot saturate all available network and storage bandwidth.

In essence, *all* of the following must be true before parallel writes are enabled:

- A single TCP connection should not saturate the available network bandwidth between the Cache and HOME server. Using multiple servers, each with their own TCP connection to the HOME servers, can make better use of the available bandwidth over a WAN.
- HOME storage aggregate write bandwidth is greater than the end-to-end network bandwidth between a single GW and HOME NFS server.
- Cache aggregate read bandwidth is larger than the end-to-end network bandwidth between a single GW and HOME NFS server.

Figure 6-11 shows that the HOME nodes are renamed for a better understanding of how to set up a parallel I/O capable environment.

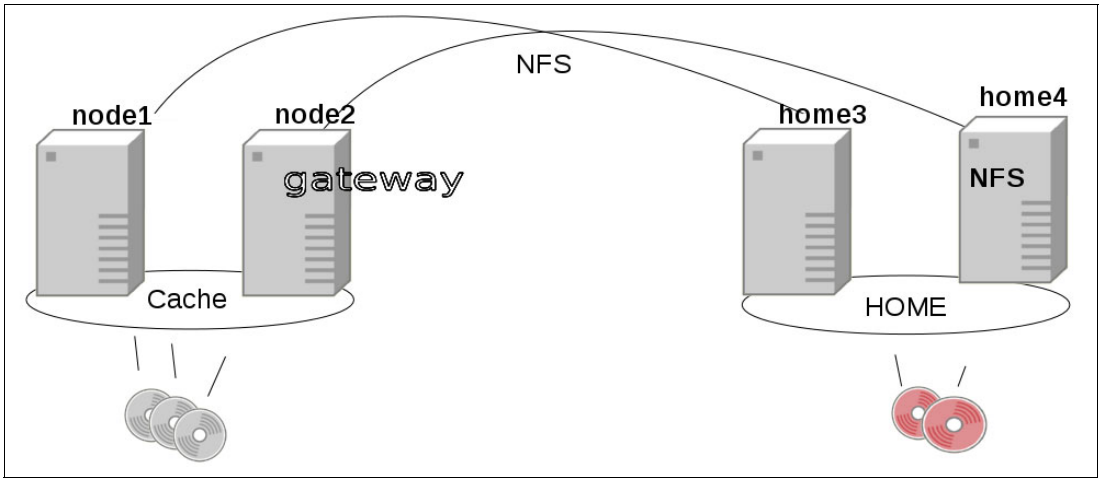


Figure 6-11 Configure AFM for parallel I/O

Example 6-67 shows an export-map has been created to define more than one HOME node for an AFM fileset using NFS.

Example 6-67 Create a mapping

```
node2:~ # mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         clusterA.site
  GPFS cluster id:           15425310929724810145
  GPFS UID domain:           clusterA.site
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address  Admin node name  Designation
-----------------------------------------------------------------
   1   node1.site        10.0.0.111  node1.site       manager-gateway
   2   node2.site        10.0.0.112  node2.site       quorum-gateway

node2:~ # mmafmconfig add beer --export-map "home3/node1,home4/node2"
mmafmconfig: Command successfully completed
mmafmconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
node2:~ # mmafmconfig show all
Map name:             beer
Export server map:    10.0.0.113/node1.site,10.0.0.114/node2.site
```

```
node2:~
```

Now that an export-map has been defined, it can be used to create the AFM fileset. You can add a map to an existing fileset by using the **failover** command to a new HOME as described in 6.2.6, “Permanent loss of HOME for single-writer” on page 306. Before creating or changing the fileset, assure that NFS is working properly. Even when using an export-map, there is a 1:1 relationship between a Cache gateway node and the NFS server so the export configuration must exist on the NFS server as shown in Example 6-68.

Example 6-68 Verify HOME’s configuration

```
### remote check

home3:~ # exportfs ; ssh home4 "exportfs"
/gpfs/fshome/fset002new
                10.0.0.0/8
/gpfs/fshome/fset002new
                10.0.0.0/8
home3:~ #

###############
```

Now you are ready to proceed and create or change the AFM fileset to use the GPFS protocol instead of NFS. Example 6-69 changes the existing fileset. The syntax that you have to specify on the AFM target is given the same way as though you would create the fileset.

Example 6-69 Create/change an AFM fileset to use new mapping

```
node2:/gpfs/cache # mmafmctl fs1 failover -j fileset_IW --new-target nfs://beer/gpfs/fshome/fset002new --target-only
mmafmctl: Performing failover to nfs://beer/gpfs/fshome/fset002new
Fileset fileset_IW changed.
node2:/gpfs/cache #

node2:/gpfs/cache/fileset_IW # mmlsfileset fs1 --afm -L
Filesets in file system 'fs1':

Attributes for fileset fileset_IW:
==================================
Status                                  Linked
Path                                    /gpfs/cache/fileset_IW
Id                                      1
Root inode                              131075
Parent Id                               0
Created                                 Fri Oct 24 01:00:57 2014
Comment
Inode space                             1
Maximum number of inodes                93184
Allocated inodes                        66560
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  nfs://beer/gpfs/fshome/fset002new
Mode                                    independent-writer
File Lookup Refresh Interval            30 (default)
File Open Refresh Interval              30 (default)
Dir Lookup Refresh Interval             60 (default)
Dir Open Refresh Interval               60 (default)
Async Delay                             15 (default)
Last pSnapId                            0
```

```
Display Home Snapshots                  no
Number of Gateway Flush Threads         4
Prefetch Threshold                      0 (default)
Eviction Enabled                        yes (default)
node2:/gpfs/cache/fileset_IW #
# verify , that it is working /active
mmafmctl fs1 getstate
Fileset Name    Fileset Target                          Cache State     Gateway Node    Queue Length   Queue numExec
------------    --------------                          -------------   ------------    ------------   -------------
fileset_IW      nfs://beer/gpfs/fshome/fset002new       Active          node2           0              324
```

The last step is optional but might be worth doing depending on your workload. There are two new tunables introduced in IBM Spectrum Scale: *afmParallel[Read,Write]Threshold*.

By customizing these tunables, you can tell IBM Spectrum Scale the clipping level parallel I/O should not be executed. The default is 1024, which means data access smaller than that cannot be transferred by parallel I/O.

You can configure these parameters as a cluster-wide tunable dynamically. To adjust these settings, see Example 6-70.

Example 6-70 Customizing parallel scaling benefits

```
node1:~ # mmchconfig afmParallelWriteThreshold=10 -i  (default 1GB)
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
node1:~ # mmchconfig afmParallelREADThreshold=10 -i  (def<ault 1 GB)
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

7

Backup and disaster recovery using IBM Spectrum Scale

This chapter describes disaster recovery (DR) configurations using IBM Spectrum Scale and the available backup tools in IBM Spectrum Scale 4.1. It also provides practical examples for implementing a typical DR configuration using IBM Spectrum Scale file system replication feature and for using the IBM Spectrum Scale backup tools and Scale Out Backup and Restore (SOBAR).

This chapter contains the followings topics:

- Disaster recovery solution using IBM Spectrum Scale replication
- Implementing a scenario with IBM Spectrum Scale replication
- Backup and restore for IBM Spectrum Scale

© Copyright IBM Corp. 2015. All rights reserved.

7.1 Disaster recovery solution using IBM Spectrum Scale replication

This section describes how IBM Spectrum Scale replication can be implemented in a disaster recovery solution.

The DR model using IBM Spectrum Scale replication consists of a single Spectrum Scale cluster that is defined across two geographic sites and a third tiebreaker site. The goal is to keep one of the sites operational in case the other site fails.

7.1.1 Configuration

This DR configuration consists of production sites (A and B) plus a tiebreaker site (C), as shown in Figure 7-1.

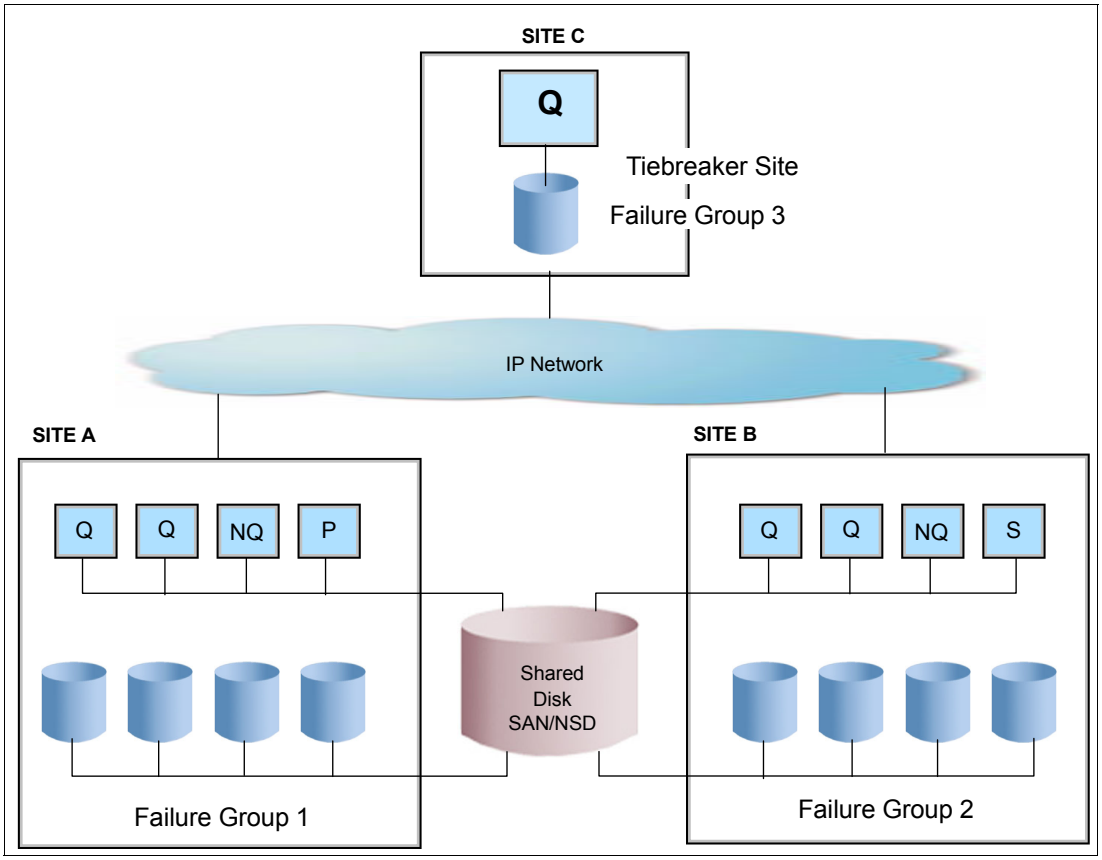


Figure 7-1 Three-site DR configuration with IBM Spectrum Scale replication

There is a single Spectrum Scale cluster containing all nodes in three sites (A, B, and C). Using the third independent site (C) as tiebreaker is essential for maintaining the node quorum and the file system quorum, in case of a site failure. Primary and secondary configurations servers are associated with nodes in primary and secondary sites (marked as P and S in Figure 7-1). These nodes can be quorum or nonquorum nodes. There are three failure groups for the defined NSDs in the three sites: one associated with each site. In case of a site down event because of the file descriptors being distributed in three sites, the file system quorum is also maintained and the file system remains mounted on the surviving site.

7.1.2 Characteristics of this DR configuration

The production sites (A and B) are used for daily production work. The tiebreaker site (C) has the following roles:

- Serve to maintain node and file system descriptor quorums after a site failure.
- Hold an additional quorum node.
- Hold a *file system descriptor-only* (descOnly) disk in a third failure group.

Additional characteristics of the DR configuration are as follows:

- Storage on the production sites is shared.
- Number of quorum nodes at each production site must be the same.
- Disks are split into two failure groups (one at each site).
- Configuration consists of one or more site-wide replicated Spectrum Scale file systems.
- Primary cluster configuration data server is configured on site A.
- Secondary cluster configuration data server is configured on site B.

Note: At the time of writing this IBM Redbooks publication, IBM Spectrum Scale 4.1.0.4 clusters using CCR are not supported for disaster recovery configurations using IBM Spectrum Scale replication. The user must disable CCR by using the **mmchcluster** command if the cluster was already enabled with CCR before configuring the cluster for replication. CCR support is intended for future IBM Spectrum Scale updates.

After a production site failure

After a production site failure, no administrative intervention is required. IBM Spectrum Scale detects the failure and reacts to it as follows:

- The failed nodes are marked as *down*.
- The failed disks are marked as *unavailable*.
- The application continues running at the surviving site.

After site recovery

Perform the following steps:

1. Restart IBM Spectrum Scale on all nodes at the recovered site:

   ```
   mmstartup -a
   ```

2. Bring the recovered disks online:

   ```
   mmchdisk fsname start
   ```

Failure on the production site and the tiebreaker site

IBM Spectrum Scale loses quorum and the file system is unmounted after the failure occurs. The administrator initiates the manual takeover procedure:

1. Change the primary configuration server in the recovery site:

   ```
   mmchcluster -p NewPrimaryServer
   ```

2. Relaxes the node quorum for the nodes in the production site:

   ```
   mmchnode --nonquorum -N NodeName
   ```

3. Relaxes the file system descriptor quorum for the disks in the production site:

   ```
   mmfsctl fsname exclude -d “Disk1;Disk2...”
   ```

7.1.3 The IBM Spectrum Scale mmfsctl command

The **mmfsctl** Spectrum Scale command implements all disaster recovery functionality:

- Use the following command before creating an IBM FlashCopy® backup. It suspends file system I/O and *flushes the Spectrum Scale cache* to ensure the integrity of the FlashCopy image:

  ```
  mmfsctl Device {suspend | suspend-write | resume}
  ```

- Use this command with active-passive storage-based replication configurations. It synchronizes the file system's configuration state between peer recovery clusters:

  ```
  mmfsctl Device syncFSconfig {-n RemoteNodesFile | -C RemoteCluster} [-S
  SpecFile]
  ```

- Use this command for minority takeover in Active-Active replicated configurations. It tells IBM Spectrum Scale to exclude the specified disks or failure groups from the file system descriptor quorum:

  ```
  mmfsctl Device {exclude | include} {-d DiskList | -F DiskFile | -G
  FailureGroup}
  ```

7.2 Implementing a scenario with IBM Spectrum Scale replication

This scenario shows a disaster recovery solution using IBM Spectrum Scale synchronous metadata and data replication between two geographically separate sites. This solution does not rely on any specific disk subsystem support.

7.2.1 Environment: Hardware, software, network, storage

The lab environment for simulation of the DR scenario contains the following hardware platforms:

- Four AIX LPARs on two IBM Power Systems; two nodes in one power system in each site (primary and secondary).
- Linux on Power LPAR (tiebreaker site).

The software is as follows:

- AIX 7.1 TL3, IBM Spectrum Scale 4.1.0.4.
- Red Hat Enterprise Linux 6.5 on Power, IBM Spectrum Scale 4.1.0.4.

On each site: The network is GbE and SAN storage is used.

7.2.2 IBM Spectrum Scale configuration

Table 7-1 on page 349 and Table 7-2 on page 349 show the configuration used for the IBM Spectrum Scale replication scenario.

Table abbreviations:

- Poughkeepsie site is abbreviated to POK.
- Kingston site is abbreviated to KGN.
- Buffalo site (tiebreaker) is abbreviated to BUF.

Table 7-1 Site: Node names, operating system images, and IP addresses

Site	Node name	Operating system	Role	IP of Spectrum Scale interfaces
POK	mexico	AIX 7.1 TL3	quorum (Primary Configuration Server)	10.11.12.121
POK	japan	AIX 7.1 TL3	quorum	10.11.12.122
KGN	france	AIX 7.1 TL3	quorum (Secondary Configuration Server)	10.11.12.123
KGN	ghana	AIX 7.1 TL3	quorum	10.11.12.124
BUF	england	RHEL 6.5	tiebreaker	10.11.12.125

Table 7-2 Site: Node names, disks, and NSD names

Site number	Site	Node name	AIX disk	NSD name
1	POK	mexico	hdisk1,hdisk2,hdisk3	mexico
	POK	japan	hdisk1,hdisk2,hdisk3	japan
2	KGN	france	hdisk4,hdisk5,hdisk6	france
	KGN	ghana	hdisk4,hdisk5,hdisk6	ghana
3	BUF	england	sdf	england

The POK and BUF sites share the SAN storage and holds the primary and secondary configuration data servers respectively. The BUF site holds a (tiebreaker) quorum node and a disk for file system descriptor only.

Notes:

- All SAN-attached disks are accessible from all nodes at sites POK and KGN.
- The NSD definitions on the disk at site BUF are defined on an internal disk that is accessible only from the BUF site.
- The tiebreaker site (BUF) serves to maintain node and file system descriptor quorum after either site POK or KGN fails.

All disks at the POK site are assigned to a failure group and all disks at the KGN site are assigned to another failure group. The disk at the BUF site is assigned to a third failure group.

A single Spectrum Scale cluster is defined across two geographic sites (Poughkeepsie and Kingston).

A single replicated Spectrum Scale file system is created from disks at the three sites.

The **mmfsctl** command (Example 7-1) is used in this scenario to manage the disaster recovery cluster:

- For minority takeover in active-active replicated configurations.
- To force IBM Spectrum Scale to exclude the specified disks or failure groups from the file system descriptor quorum.

Example 7-1 The mmfsctl command syntax

```
mmfsctl Device {exclude | include} {-d DiskList | -F DiskFile | -G FailureGroup}
```

7.2.3 IBM Spectrum Scale configuration diagram

Figure 7-2 shows a diagram of the configuration. Table 7-3 lists the meanings for the letter designations in the figure.

Table 7-3 Letter designation and meanings for Figure 7-2

Letter	Role
Q	Quorum
NQ	Nonquorum
P	Primary configuration data server
S	Secondary configuration data server

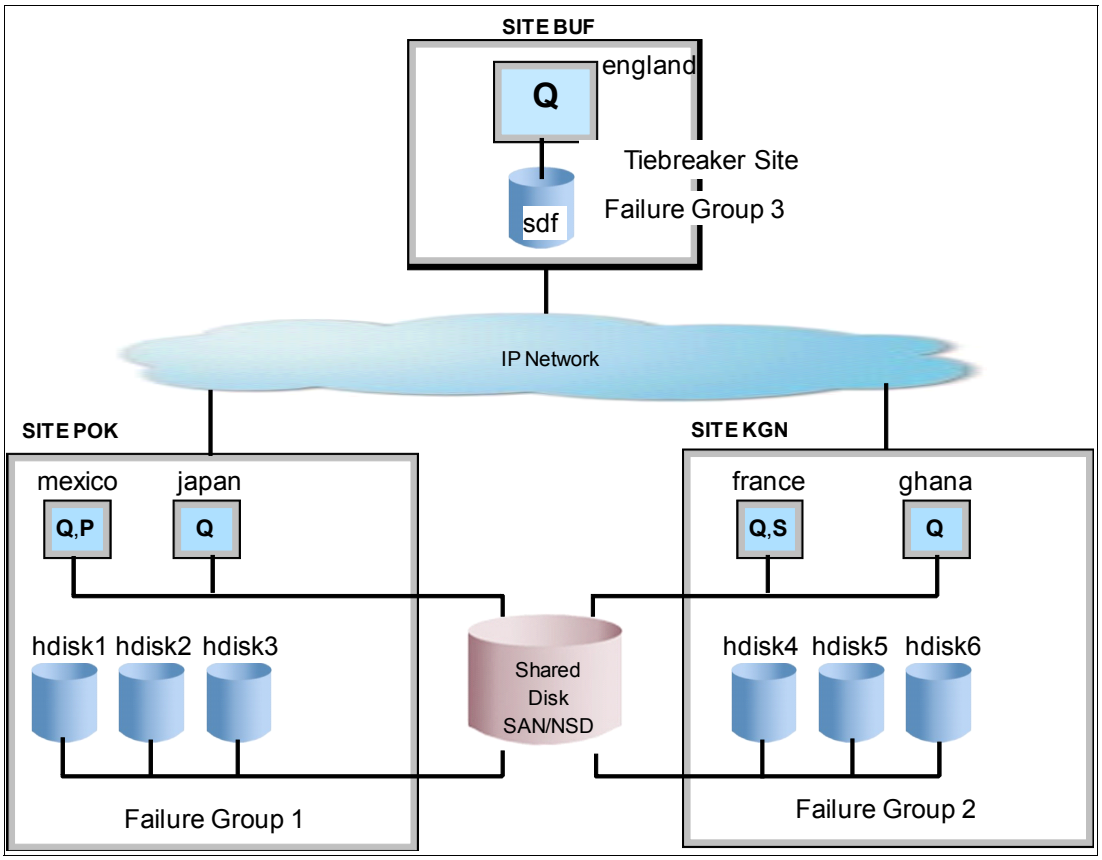


Figure 7-2 Three-site IBM Spectrum Scale DR cluster and associated shared disks

7.2.4 Set up and configure the IBM Spectrum Scale DR cluster

The following section provides details about how to set up and configure the Spectrum Scale disaster recovery cluster.

The steps are as follows:

1. Set up node description file for creating the Spectrum Scale cluster. Create a node description file for defining the Spectrum Scale cluster across all three sites. See Example 7-2.

Example 7-2 Create a node description file for use by the mmcrcluster command

```
mexico:quorum-manager:
japan:quorum-manager:
france:quorum-manager:
ghana:quorum-manager:
england:quorum-client:
```

Note: You must define the node on BUF site as a quorum and client, not a manager.

2. Create the Spectrum Scale cluster. Issue the **mmcrcluster** command with the node description file as shown in Example 7-2 to create the Spectrum Scale cluster as shown in Example 7-3.

Example 7-3 Use mmcrcluster with the previous input file to create the Spectrum Scale cluster

```
[root on mexico][/tmp] => /usr/lpp/mmfs/bin/mmcrcluster -N /tmp/Node_Desc_File1
--ccr-disable -p mexico -s france -r /usr/bin/ssh -R /usr/bin/scp
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmcrcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root on mexico][/tmp] =>
```

3. List the node of the Spectrum Scale cluster. Display the Spectrum Scale cluster definition with the **mmlscluster** command. See Example 7-4.

Example 7-4 Use the mmlscluster command to display details of the cluster

```
[root on mexico][/tmp] => mmlscluster

===============================================================================
| Warning:                                                                    |
|   This cluster contains nodes that do not have a proper GPFS license        |
|   designation.  This violates the terms of the GPFS licensing agreement.    |
|   Use the mmchlicense command and assign the appropriate GPFS licenses      |
|   to each of the nodes in the cluster.  For more information about GPFS     |
|   license designation, see the Concepts, Planning, and Installation Guide.  |
===============================================================================


GPFS cluster information
```

```
=======================
  GPFS cluster name:         mexico
  GPFS cluster id:           13882458522525678021
  GPFS UID domain:           mexico
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           server-based

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    mexico
  Secondary server:  france

 Node  Daemon node name            IP address       Admin node name             Designation
-----------------------------------------------------------------------------------------------
   1   mexico                      10.11.12.121     mexico                      quorum-manager
   2   japan                       10.11.12.122     japan                       quorum-manager
   3   france                      10.11.12.123     france                      quorum-manager
   4   ghana                       10.11.12.124     ghana                       quorum-manager
   5   england                     10.11.12.125     england                     quorum

[root on mexico][/tmp] =>
```

4. List the Spectrum Scale server licenses on the cluster. Show server licenses of all nodes in the cluster by using the **mmlslicense** command. See Example 7-5.

Example 7-5 Use mmlslicense to display licenses defined for each node in the cluster

```
[root on mexico][/tmp] => mmlslicense -L
 Node name                     Required license   Designated license
--------------------------------------------------------------------
mexico                            server            none    *
japan                             server            none    *
france                            server            none    *
ghana                             server            none    *
england                           server            none    *

 Summary information
---------------------
Number of nodes defined in the cluster:                         5
Number of nodes with server license designation:                0
Number of nodes with client license designation:                0
Number of nodes still requiring server license designation:     5
Number of nodes still requiring client license designation:     0
[root on mexico][/tmp] =>
```

5. Change the server licenses to the nodes in the cluster. See Example 7-6.

Example 7-6 Use mmchlicense to assign server licenses

```
[root on mexico][/tmp] => mmchlicense server --accept -N all

The following nodes will be designated as possessing GPFS server licenses:
        mexico
        japan
        france
        ghana
        england
mmchlicense: Command successfully completed
```

```
mmchlicense: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on mexico][/tmp] =>
```

6. Check the server licenses of all nodes in the cluster. See Example 7-7.

Example 7-7 Use mmlslicense to display the current licenses of all nodes in the cluster

```
[root on mexico][/tmp] => mmlslicense -L
 Node name                    Required license   Designated license
--------------------------------------------------------------------
mexico                             server            server
japan                              server            server
france                             server            server
ghana                              server            server
england                            server            server

 Summary information
---------------------
Number of nodes defined in the cluster:                         5
Number of nodes with server license designation:                5
Number of nodes with client license designation:                0
Number of nodes still requiring server license designation:     0
Number of nodes still requiring client license designation:     0
[root on mexico][/tmp] =>
```

7. Set up the Spectrum Scale cluster configuration parameters specific to your environment by using the **mmchconfig** command. See Example 7-8.

Example 7-8 Use the mmchconfig command to set up specific Spectrum Scale cluster parameters

```
[root on mexico][/] => mmchconfig maxblocksize=4M
Verifying GPFS is stopped on all nodes ...
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

[root on mexico][/] => mmlsconfig
Configuration data for cluster mexico:
--------------------------------------
clusterName mexico
clusterId 13882458522525678021
autoload no
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
maxblocksize 4M
adminMode central

File systems in cluster mexico:
-------------------------------
(none)
[root on mexico][/] =>
```

8. Define NSDs for the Spectrum Scale cluster. Set up an NSD descriptor file to contain disks from all three sites (POK, KGN, and BUF). Disks at sites POK and KGN are assigned to failure group 3001 and 3002 respectively. The single disk at the BUF site is assigned to failure group 3003. See Example 7-9.

Example 7-9 Create an NSD descriptor file to be used as input to the mmcrnsd command

```
%nsd: device=hdisk1 servers=mexico,japan usage=dataAndMetadata failuregroup=3001
%nsd: device=hdisk2 servers=japan,mexico usage=dataAndMetadata failuregroup=3001
%nsd: device=hdisk3 servers=mexico,japan usage=dataAndMetadata failuregroup=3001
%nsd: device=hdisk4 servers=france,ghana usage=dataAndMetadata failuregroup=3002
%nsd: device=hdisk5 servers=ghana,france usage=dataAndMetadata failuregroup=3002
%nsd: device=hdisk6 servers=france,ghana usage=dataAndMetadata failuregroup=3002
%nsd: device=sdf servers=england usage=descOnly failuregroup=3003
```

9. Create NSDs on the Spectrum Scale cluster. Use the **mmcrnsd** command to create NSDs. See Example 7-10.

Example 7-10 Create NSDs for this cluster with the mmcrnsd command

```
[root on mexico][/tmp] => mmcrnsd -F NSD_Desc_File1
mmcrnsd: Processing disk hdisk1
mmcrnsd: Processing disk hdisk2
mmcrnsd: Processing disk hdisk3
mmcrnsd: Processing disk hdisk4
mmcrnsd: Processing disk hdisk5
mmcrnsd: Processing disk hdisk6
mmcrnsd: Processing disk sdf
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on mexico][/tmp] =>
```

10. Check NSD on the cluster. Display defined NSDs for this cluster by using the **mmlsnsd** command. See Example 7-11.

Example 7-11 Use mmlsnsd to display all NSDs on the cluster

```
[root on mexico][/tmp] => mmlsnsd

File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   gpfs1nsd     mexico,japan
 (free disk)   gpfs2nsd     japan,mexico
 (free disk)   gpfs3nsd     mexico,japan
 (free disk)   gpfs4nsd     france,ghana
 (free disk)   gpfs5nsd     ghana,france
 (free disk)   gpfs6nsd     france,ghana
 (free disk)   gpfs7nsd     england

[root on mexico][/tmp] =>
```

11. Create a site-wide Spectrum Scale cluster. Define a Spectrum Scale cluster over three geographically separate sites. Two sites are production sites and the third site is a tiebreaker site. File systems are mounted on all three nodes and are accessed concurrently from both production sites. The disk subsystem is shared between the production sites and the disk on the tiebreaker site is a local internal disk. Before creating the file system startup, the Spectrum Scale services on all nodes using: **mmstartup -a**.

In our scenario, we create the site-wide Spectrum Scale file system named gpfs1. Use the **mmcrfs** command to create the file system. See Example 7-12.

Example 7-12 Use mmcrfs to create site-wide file system

```
mmcrfs gpfs1 -F NSD_Desc_File1 -r 2 -R 2 -m 2 -M 2 -T /gpfsSH1 -A yes
```

12. Example 7-13 shows the output from the **mmcrfs** command.

Example 7-13 Output from mmcrfs command

```
[root on mexico][/tmp] =>  mmcrfs_gpfs1.sh

The following disks of gpfs1 will be formatted on node mexico:
    gpfs1nsd: size 26214400 KB
    gpfs2nsd: size 26214400 KB
    gpfs3nsd: size 26214400 KB
    gpfs4nsd: size 26214400 KB
    gpfs5nsd: size 26214400 KB
    gpfs6nsd: size 26214400 KB
    gpfs7nsd: size 5242880 KB
Formatting file system ...
Disks up to size 409 GB can be added to storage pool 'system'.
Creating Inode File
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on mexico][/tmp] =>
```

13. Verify the file system defined to the Spectrum Scale cluster. Use the **mmlsconfig** command to list the system. See Example 7-14.

Example 7-14 Use mmlsconfig to display the Spectrum Scale file system on this cluster

```
[root on mexico][/tmp] => mmlsconfig
Configuration data for cluster mexico:
---------------------------------------
clusterName mexico
clusterId 13882458522525678021
autoload no
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
maxblocksize 4M
adminMode central

File systems in cluster mexico:
-------------------------------
/dev/gpfs1
[root on mexico][/tmp] =>
```

14. Mount all Spectrum Scale file systems on the cluster. Use the **mmmount** command to mount the systems. See Example 7-15.

Example 7-15 Use mmmount all -a to mount all defined file systems on all nodes of the cluster

```
[root on mexico][/tmp] => mmmount all -a
Wed Nov 5 18:22:15 EDT 2014: mmmount: Mounting file systems ...
[root on mexico][/tmp] => mmlsmount all
File system gpfs1 is mounted on 5 nodes.
[root on mexico][/tmp] => mmlsmount all -L

File system gpfs1 is mounted on 5 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.123    france
  10.11.12.124    ghana
  10.11.12.125    england
[root on mexico][/tmp] =>

[root on mexico][/tmp] => df  | grep gpfs
/dev/gpfs1     314572800 313642496    1%     4070     3% /gpfsSH1
[root on mexico][/tmp] =>
```

15. Verify that the Spectrum Scale daemon is active on all cluster nodes. Use the **mmgetstate** command to get the status. See Example 7-16.

Example 7-16 Use mmgetstate to check the status of all Spectrum Scale daemons

```
[root@england ~]# mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico             3        5          5       active      quorum node
       2      japan              3        5          5       active      quorum node
       3      france             3        5          5       active      quorum node
       4      ghana              3        5          5       active      quorum node
       5      england            3        5          5       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       5
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      5
Quorum = 3, Quorum achieved

[root@england ~]#
```

16. Check the state of all disks in the file system. Verify that all disks that are defined to the Spectrum Scale file system are in “up” state. See Example 7-17.

Example 7-17 Check the state of all disks in the file systems

```
[root@england ~]# mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
```

```
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs5nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs6nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs7nsd     nsd         512    3003 no       no    ready         up           system
[root@england ~]#
```

17. Prevent false disk errors in the SAN configuration from being reported to the file system manager by enabling the **unmountOnDiskFail** option on the tiebreaker node. See Example 7-18.

Example 7-18 Set unmountOnDiskFail=yes with the mmchconfig command

```
[root@england ~]# mmchconfig unmountOnDiskFail=yes -N england
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root@england ~]#
```

18. On each node of the production sites (KGN, POK), start a stress workload to run for an extended period. As an example, the **dd** command can be used to write data to the Spectrum Scale file system.

Fail over after failure of KGN site

To simulate the complete failure of the KGN site, the Spectrum Scale daemon on both (france and ghana) is shut down. Furthermore, we take down the storage subsystem that is associated with NSD at that site (france and ghana).

Important: After a site failure, the following events occur:

1. IBM Spectrum Scale detects and responds to the failure.
2. The failed nodes are marked as “down.”
3. The failed disks are marked as “down.”

All applications on the surviving site (POK) continue to run without errors.

The steps are as follows:

1. Shut down IBM Spectrum Scale on france and ghana. From the command line, run the **/usr/lpp/mmfs/bin/mmshutdown** command on france and ghana. See Example 7-19.

Example 7-19 Shutdown IBM Spectrum Scale on france and ghana with mmshutdown command

```
[root on france][/tmp] => mmshutdown
Thu Nov 6 00:13:57 EDT 2014: mmshutdown: Starting force unmount of GPFS file
systems
forced unmount of /gpfsSH1
Thu Nov 6 00:14:02 EDT 2014: mmshutdown: Shutting down GPFS daemons
Shutting down!
'shutdown' command about to kill process 7798924
Thu Nov 6 00:14:07 EDT 2014: mmshutdown: Finished
[root on france][/tmp] =>

[root on ghana][/tmp] => mmshutdown
Thu Nov 6 00:14:02 EDT 2014: mmshutdown: Starting force unmount of GPFS file
systems
forced unmount of /gpfsSH1
```

```
Thu Nov 6 00:14:07 EDT 2014: mmshutdown: Shutting down GPFS daemons
Shutting down!
'shutdown' command about to kill process 8716302
Thu Nov 6 00:15:12 EDT 2014: mmshutdown: Finished
[root on ghana][/tmp] =>
```

2. Verify that the Spectrum Scale daemon is down on france and ghana. Use the **mmgetstate** command to verify the daemon. See Example 7-20.

Example 7-20 Verify Spectrum Scale daemon is down on france and ghana; use mmgetstate

```
[root on ghana][/tmp] => mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico             3        3          5       active      quorum node
       2      japan              3        3          5       active      quorum node
       3      france             0        0          5       down        quorum node
       4      ghana              0        0          5       down        quorum node
       5      england            3        3          5       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       3
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      3
Quorum = 3, Quorum achieved

[root on ghana][/tmp] =>
```

3. Verify that file systems are mounted on the surviving sites. After the KGN site fails, the Spectrum Scale file system (/gpfsSH1) must remain mounted and accessible on all nodes of the POK site and the tiebreaker site. Use **mmlsmount**. See Example 7-21.

Example 7-21 Verify gpfs1 file system is mounted on all surviving nodes

```
[root on ghana][/tmp] => mmlsmount all -L

File system gpfs1 is mounted on 3 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.125    england
[root on ghana][/tmp] =>
```

4. Verify disks served by the KGN site are “down.” After the failure of the KGN site, the disk server at that site should be unavailable. Use **mmlsdisk** to verify. See Example 7-22.

Example 7-22 Disks server by nodes at site KGN are in down state

```
[root@england ~]# mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         down         system
gpfs5nsd     nsd         512    3002 yes      yes   ready         down         system
```

```
gpfs6nsd     nsd         512    3002 yes      yes   ready         down          system
gpfs7nsd     nsd         512    3003 no       no    ready         up            system
[root@england ~]#
```

5. Verify that workload running on the POK site continues to run error-free. After the failure of the KGN site, the nodes and disk on the POK site remain available, and any work currently running on the nodes at the POK site continue to run uninterrupted. See Example 7-23.

Example 7-23 Workload running at POK site continues to run error-free

```
.02    /gpfsSH1/japan477/samples/util
0.77    /gpfsSH1/japan477/samples
130.69  /gpfsSH1/japan477
0.05    /gpfsSH1/japan478/README$
0.36    /gpfsSH1/japan478/bin/.links
49.70   /gpfsSH1/japan478/bin/aix32
52.53   /gpfsSH1/japan478/bin/aix64
127.61  /gpfsSH1/japan478/bin
0.12    /gpfsSH1/japan478/data
0.42    /gpfsSH1/japan478/include
0.48    /gpfsSH1/japan478/lib
0.44    /gpfsSH1/japan478/messages
0.36    /gpfsSH1/japan478/samples/debugtools
0.16    /gpfsSH1/japan478/samples/ilm
0.02    /gpfsSH1/japan478/samples/net
0.02    /gpfsSH1/japan478/samples/perf
0.02    /gpfsSH1/japan478/samples/uidremap
0.02    /gpfsSH1/japan478/samples/util
0.77    /gpfsSH1/japan478/samples
133.05  /gpfsSH1/japan478
```

6. Start the Spectrum Scale daemon on all nodes of the KGN site with the **mmstartup** command. See Example 7-24.

Example 7-24 Run the mmstartup command to restart the Spectrum Scale daemons at site KGN

```
[root on france][/tmp] => mmstartup
Thu Nov 6 00:46:46 EDT 2014: mmstartup: Starting GPFS ...
[root on france][/tmp] =>

[root on ghana][/tmp] => mmstartup
Thu Nov 6 00:46:51 EDT 2014: mmstartup: Starting GPFS ...
[root on ghana][/tmp] =>
```

7. Verify that the Spectrum Scale daemons are restarted on all nodes of the KGN site. Use **mmgetstate** to verify the daemons. See Example 7-25.

Example 7-25 Use mmgetstate to check the state of the Spectrum Scale daemon at all sites

```
[root@england ~]# mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico            3        5          5       active      quorum node
       2      japan             3        5          5       active      quorum node
       3      france            3        5          5       active      quorum node
       4      ghana             3        5          5       active      quorum node
       5      england           3        5          5       active      quorum node
```

```
 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       5
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      5
Quorum = 3, Quorum achieved

[root@england ~]#
```

8. Verify status of the disks in the file systems. After the Spectrum Scale daemon on the nodes at the KGN site is started and the file system is mounted, the NSD from the KGN site remains in the down state. See Example 7-26.

Example 7-26 NSD from site KGN is in the down state after the daemon starts

```
[root@england ~]# mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         down         system
gpfs5nsd     nsd         512    3002 yes      yes   ready         down         system
gpfs6nsd     nsd         512    3002 yes      yes   ready         down         system
gpfs7nsd     nsd         512    3003 no       no    ready         up           system
[root@england ~]#
```

9. The NSDs that are served by the nodes at the KGN site must be brought back online (up) manually with the **mmchdisk** command. See Example 7-27.

Example 7-27 mmchdisk command used to bring "down" disk back up

```
[root on ghana][/tmp] => mmchdisk /dev/gpfs1 start -d "gpfs6nsd;gpfs5nsd;gpfs4nsd"
mmnsddiscover:  Attempting to rediscover the disks.  This may take a while ...
mmnsddiscover:  Finished.
ghana:  Rediscovered nsd server access to gpfs6nsd.
ghana:  Rediscovered nsd server access to gpfs5nsd.
ghana:  Rediscovered nsd server access to gpfs4nsd.
Scanning file system metadata, phase 1 ...
  59 % complete on Thu Nov 6 00:53:32 2014
 100 % complete on Thu Nov 6 00:53:34 2014
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 33.45 % complete on Thu Nov 6 00:53:55 2014    (    439448 inodes      52408 MB)
 38.52 % complete on Thu Nov 6 00:54:15 2014    (    508205 inodes      60353 MB)
 42.81 % complete on Thu Nov 6 00:54:36 2014    (    589497 inodes      67077 MB)
 45.71 % complete on Thu Nov 6 00:54:56 2014    (    643186 inodes      71617 MB)
 47.11 % complete on Thu Nov 6 00:55:17 2014    (    667588 inodes      73825 MB)
 48.86 % complete on Thu Nov 6 00:55:38 2014    (    692211 inodes      76564 MB)
 50.38 % complete on Thu Nov 6 00:55:58 2014    (    741528 inodes      78944 MB)
```

```
 100.00 % complete on Thu Nov 6 00:56:14 2014
Scan completed successfully.
[root on ghana][/tmp] =>
```

10. Verify that all down NSDs at the KGN site are now available. Use the **mmlsdisk** command to verify that the status availability is in the “up” state. See Example 7-28.

Example 7-28 Use mmlsdisk to verify disks in file system gpfs1 are now up

```
[root on ghana][/tmp] => mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs5nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs6nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs7nsd     nsd         512    3003 no       no    ready         up           system
[root on ghana][/tmp] =>
```

11. Rebalance the /gpfs1 file system. Use the **mmrestripefs** command with the **-b** option to rebalance all files for all disks. See Example 7-29.

Example 7-29 mmrestripefs rebalances the file system and restores the replication of all files

```
[root on ghana][/tmp] => mmrestripefs gpfs1 -b  -N all
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 0.44 % complete on Thu Nov 6 01:01:40 2014     (    204572 inodes        370 MB)
 0.62 % complete on Thu Nov 6 01:02:00 2014     (    505332 inodes        519 MB)
 100.00 % complete on Thu Nov 6 01:02:07 2014
Scan completed successfully.
[root on ghana][/tmp] =>
```

12. Verify that gpfs1 is mounted on all nodes at all sites. Use the **mmlsmount** command to verify. See Example 7-30.

Example 7-30 The gpfs1 is mounted on all nodes at all sites after the KGN site recovers

```
[root on ghana][/tmp] => mmlsmount all -L

File system gpfs1 is mounted on 5 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.125    england
  10.11.12.123    france
  10.11.12.124    ghana
[root on ghana][/tmp] =>
```

Unattended stop of Spectrum Scale nodes in site KGN

To ensure that the correct LPAR is shut down, identify the name of the LPAR that is associated with the nodes at the KGN site.

One way to identify the LPAR associated with a particular node is as follows:

1. Log in to the node on the site (KGN) for which you want to identify its LPAR.
2. Run the **lparstat -i** command to display the partition name and node name. See Example 7-31.
3. Obtain the partition name associated with ghana.

Example 7-31 Use lparstat -i to identify the partition name associated with this host name

```
[root on ghana][/] => lparstat -i
Node Name                                  : ghana
Partition Name                             : 550_3_LP04
Partition Number                           : 6
Type                                       : Shared-SMT
Mode                                       : Capped
Entitled Capacity                          : 0.50
Partition Group-ID                         : 32774
Shared Pool ID                             : 0
Online Virtual CPUs                        : 2
Maximum Virtual CPUs                       : 4
Minimum Virtual CPUs                       : 1
Online Memory                              : 4224 MB
Maximum Memory                             : 5248 MB
Minimum Memory                             : 1152 MB
Variable Capacity Weight                   : 0
Minimum Capacity                           : 0.10
Maximum Capacity                           : 4.00
Capacity Increment                         : 0.01
Maximum Physical CPUs in system            : 4
Active Physical CPUs in system             : 4
Active CPUs in Pool                        : 4
Shared Physical CPUs in system             : 4
Maximum Capacity of Pool                   : 400
Entitled Capacity of Pool                  : 350
Unallocated Capacity                       : 0.00
Physical CPU Percentage                    : 25.00%
Unallocated Weight                         : 0
Memory Mode                                : Dedicated
Total I/O Memory Entitlement               : -
Variable Memory Capacity Weight            : -
Memory Pool ID                             : -
Physical Memory in the Pool                : -
Hypervisor Page Size                       : -
Unallocated Variable Memory Capacity Weight: -
Unallocated I/O Memory entitlement         : -
Memory Group ID of LPAR                    : -
Desired Virtual CPUs                       : 2
Desired Memory                             : 4224 MB
Desired Variable Capacity Weight           : 0
Desired Capacity                           : 0.50
Target Memory Expansion Factor             : -
Target Memory Expansion Size               : -
```

```
[root on ghana][/] =>
```

4. Shut down LPARs associated with france and ghana. Log in to the HMC console that is associated with france and ghana, and then perform the LPAR shutdown operation. See Figure 7-3.

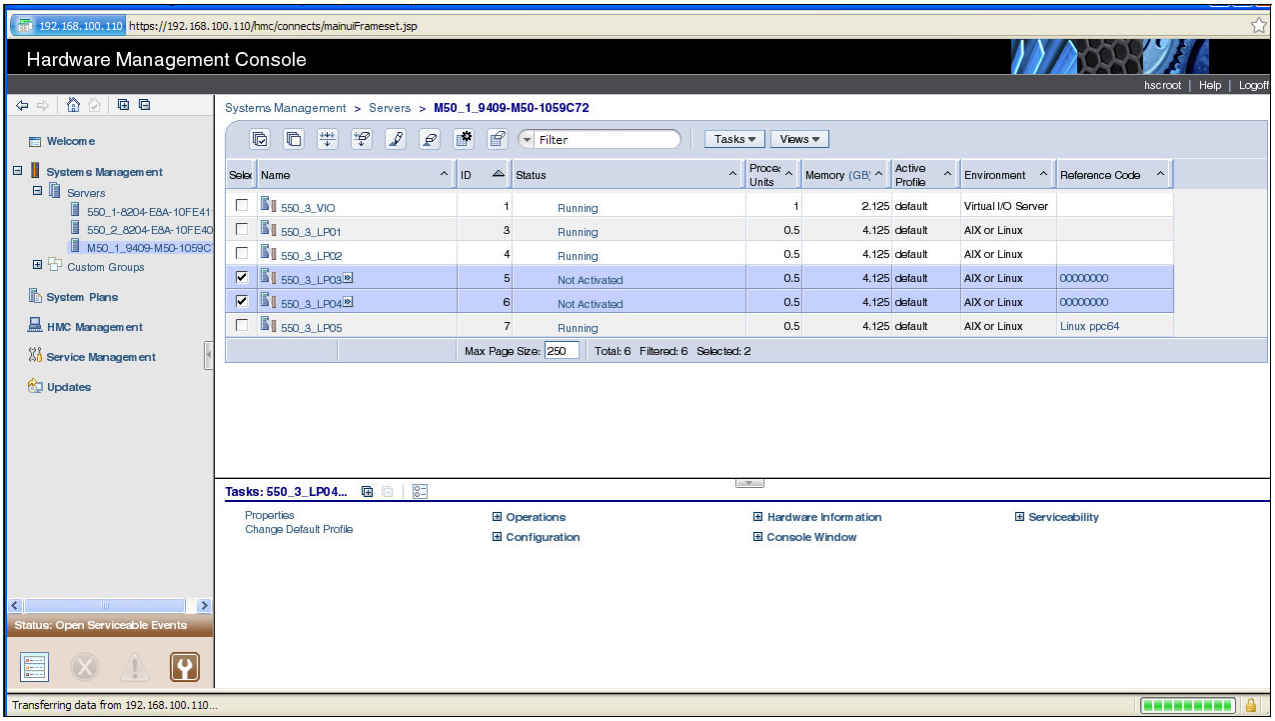


Figure 7-3 HMC console that is used to shut down france and ghana

5. Verify that france and ghana LPARs are shut down. Use the **mmgetstate command** to verify the status. See Example 7-32.

Example 7-32 Check the state of france and ghana with the mmgetstate command

```
[root@england ~]# mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico             3        3          5       active      quorum node
       2      japan              3        3          5       active      quorum node
       3      france             0        0          5       unknown     quorum node
       4      ghana              0        0          5       unknown     quorum node
       5      england            3        3          5       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       3
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      3
Quorum = 3, Quorum achieved

[root@england ~]#
```

6. Verify that gpfs1 is mounted at the POK site after site shutdown. Use the **mmlsmount** command to determine the status of /gpfs1 on the surviving site. See Example 7-33.

Example 7-33 Use mmlsmount to check the mount status of gpfs1 on the surviving site (POK)

```
[root on japan][/] => mmlsmount all -L

File system gpfs1 is mounted on 3 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.125    england
[root on japan][/] =>
```

7. Verify gpfs1 is accessible from all nodes of the POK site. Verify that the workload, which is running on the nodes of the POK site, is running without errors. See Example 7-34.

Example 7-34 Verify workload running on the nodes of the POK site run error free

```
[root on mexico][/] => du -m /gpfsSH1/japan0
0.05    /gpfsSH1/japan0/README$
0.36    /gpfsSH1/japan0/bin/.links
50.70   /gpfsSH1/japan0/bin/aix32
54.97   /gpfsSH1/japan0/bin/aix64
131.55  /gpfsSH1/japan0/bin
2.25    /gpfsSH1/japan0/samples
138.47  /gpfsSH1/japan0
[root on mexico][/] =>

[root on japan][/] =>du -m /gpfsSH1/france0
1166.47 /gpfsSH1/france0/mexico0/bos/inst_root
1374.09 /gpfsSH1/france0/mexico0/bos
0.38    /gpfsSH1/france0/mexico0/samples/perf
0.05    /gpfsSH1/france0/mexico0/samples/uidremap
0.67    /gpfsSH1/france0/mexico0/samples/util
2.41    /gpfsSH1/france0/mexico0/samples
2316.59 /gpfsSH1/france0/mexico0
2316.61 /gpfsSH1/france0
[root on japan][/] =>
```

Important: While the Spectrum Scale cluster is in a failover state, no changes to the Spectrum Scale configuration are made.

8. Reactivate the LPARs for france and ghana. Log in to the HMC that manages the nodes (france and ghana) and reactivate these LPARs, as shown in Figure 7-4 on page 365.

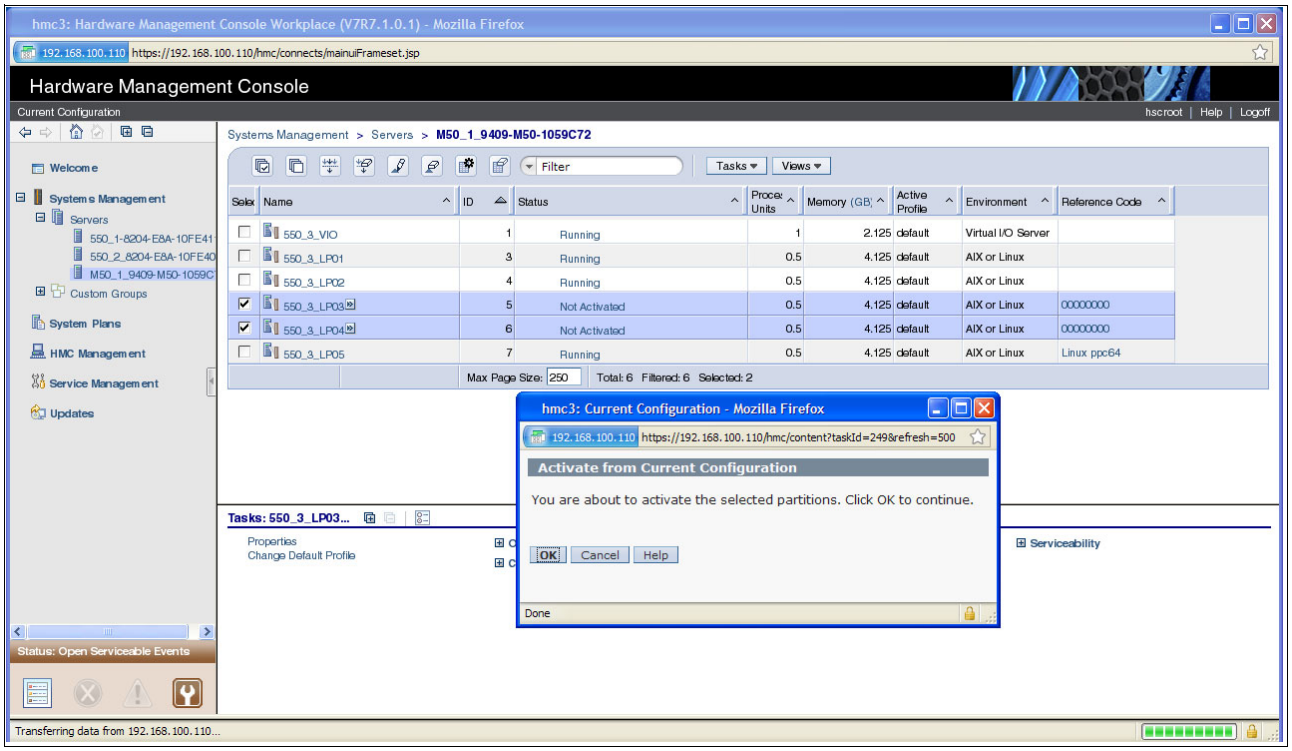


Figure 7-4 HMC console during activation of the france and ghana LPARs

After reactivation of the france and ghana LPARs, the HMC console looks similar to Figure 7-5.

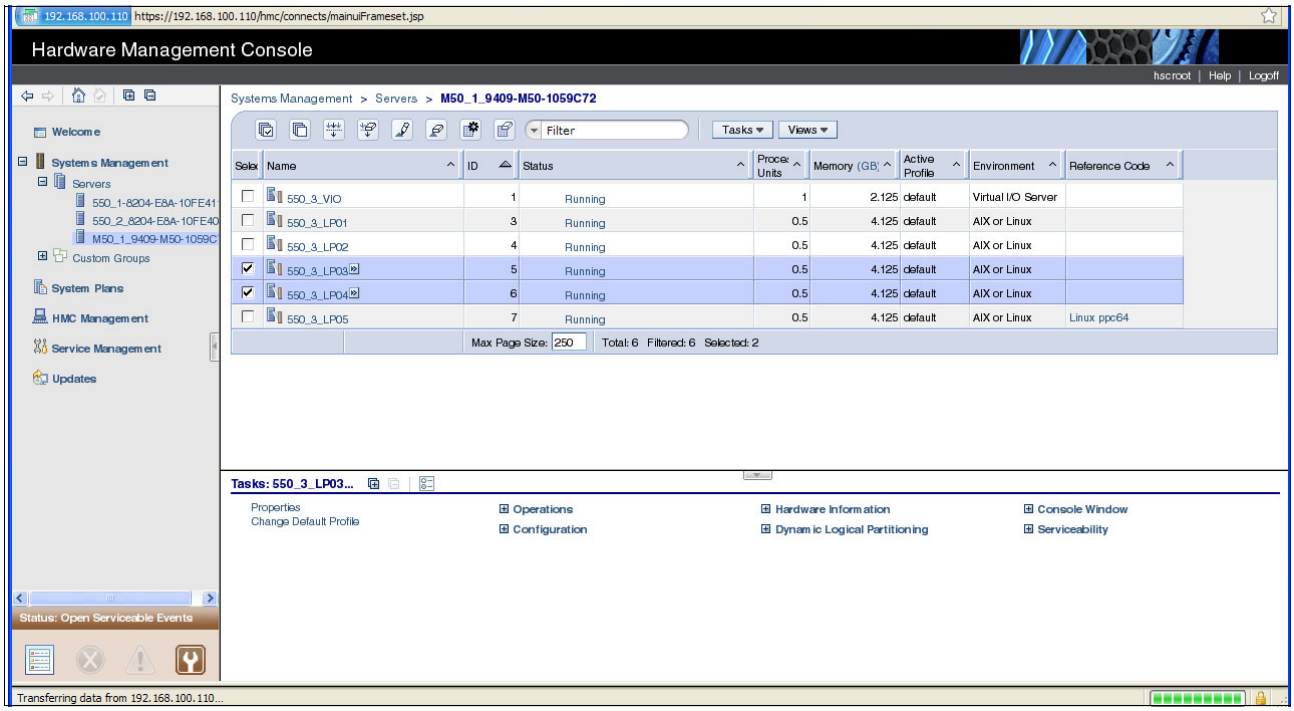


Figure 7-5 HMC console after reactivation of france and ghana LPARs

9. Start IBM Spectrum Scale on all nodes of the KGN site. After the LPARs for france and ghana are reactivated, start the Spectrum Scale daemon on these LPARs. See Example 7-35 on page 366.

Example 7-35 Manually start the Spectrum Scale daemon on all nodes of the KGN site

```
[root on france][/] => mmstartup
Thu Nov 6 13:12:07 EDT 2014: mmstartup: Starting GPFS ...
[root on france][/] =>

[root on ghana][/] => mmstartup
Thu Nov 6 13:12:32 EDT 2014: mmstartup: Starting GPFS ...
[root on ghana][/] =>
```

10. Verify that the Spectrum Scale daemon is restarted on all nodes of the KGN site. Use the **mmgetstate** command to verify the active state. See Example 7-36.

Example 7-36 Use mmgetstate to check status of the Spectrum Scale daemon on all nodes

```
[root on mexico][/] => mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico             3        5          5       active      quorum node
       2      japan              3        5          5       active      quorum node
       3      france             3        5          5       active      quorum node
       4      ghana              3        5          5       active      quorum node
       5      england            3        5          5       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       5
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      5
Quorum = 3, Quorum achieved

[root on mexico][/] =>
```

11. Verify that gpfs1 is remounted on all nodes of all sites. Use the **mmlsmount** command to verify that the file system is mounted. See Example 7-37.

Example 7-37 Use mmlsmount to verify gpfs1 is mounted on all nodes of all sites

```
[root on mexico][/] => mmlsmount all -L

File system gpfs1 is mounted on 5 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.125    england
  10.11.12.123    france
  10.11.12.124    ghana
[root on mexico][/] =>
```

12. Start the disks part of the gpfs1 file system on KNG site using command **mmchdisk start** and restripe of the file system using **mmrestripefs**, as shown in Example 7-27 on page 360 and respectively in, Example 7-29 on page 361. Verify the status of all disks in

the file systems. Use the **mmlsdisk** command to verify the status. See Example 7-38 on page 367.

Example 7-38 Verify the status of each disk in the file system

```
[root on mexico][/] => mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs5nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs6nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs7nsd     nsd         512    3003 no       no    ready         up           system
[root on mexico][/] =>
```

Fail over to surviving site: tiebreaker affected

This section describes how to recover from the failure of two sites in a three-site disaster recovery configuration where the tiebreaker site is also affected. If the tiebreaker site is no longer operational after a disaster, the Spectrum Scale node and file system quorum is broken. When the Spectrum Scale quorum is broken, manual intervention is required to resume file system access. For the surviving site to satisfy its quorum requirements and remain operational, the existing quorum configuration must be modified.

The following changes are made for the surviving site to continue:

1. If running in DR case takes a longer time and additional changes to the cluster configuration are required, reassign the primary configuration server to a node in the surviving site using: **mmchcluster -p <newnodeDR>**.
2. Relax node quorum requirements.

   To relax node quorum, use the **mmchnode --nonquorum** command, which temporarily changes the designation of each of the failed quorum nodes in the failed sites to nonquorum nodes.
3. Relax file system descriptor quorum.

   To relax file descriptor quorum, use the **mmfsctl exclude** command to temporarily eliminate the failed disks from the group of disks from which the Spectrum Scale daemon uses to write the file system descriptor file.

Steps performed in our scenario:

1. Power down all nodes at site POK - mexico and japan. Use HMC to identify the LPARs associated with mexico and japan. Power down the LPAR associated with mexico and japan using shutdown immediately and disconnect the disks from POK so that site KGN cannot have direct access using the SAN. Then, power down the LPAR associated with the BUF site.

2. Verify that all nodes are down at the POK and BUF sites. After all nodes at the site (POK) are powered off, IBM Spectrum Scale detects the failure and goes into “arbitrating” state because node quorum has been lost. See Example 7-39.

Example 7-39 Use mmgetstate to verify all nodes at site (KGN) are down

```
[root on france][/] => mmgetstate -aL
```

```
 Node number  Node name       Quorum  Nodes up  Total nodes  GFPS state  Remarks
------------------------------------------------------------------------------------
       1      mexico            0       0          5       unknown     quorum node
       2      japan             0       0          5       unknown     quorum node
       3      france            3       0          5       arbitrating quorum node
       4      ghana             3       0          5       arbitrating quorum node
       5      england           0       0          5       unknown     quorum node

[root on france][/] =>
```

3. Check the state of file systems on the cluster. The **mmlsmount** command fails because the nodes are down. See Example 7-40.

Example 7-40 The mmlsmount command fails because all nodes in site KGN and BUF are down

```
[root on france][/] => mmlsmount all -L
Device not ready.
mmlsmount: Command was unable to determine whether file system gpfs1 is mounted.
mmlsmount: Command failed.  Examine previous error messages to determine cause.
[root on france][/]

[root on france][/] => df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          655360     95496   86%    14178    52% /
/dev/hd2         6029312     79088   99%    48134    77% /usr
/dev/hd9var      2752512   2079288   25%    11571     5% /var
/dev/hd3         2359296   2050080   14%      125     1% /tmp
/dev/hd1          131072    130360    1%        5     1% /home
/dev/hd11admin     131072    130360    1%        5     1% /admin
/proc                 -         -    -         -     -  /proc
/dev/hd10opt      524288    119888   78%     8618    39% /opt
/dev/livedump     131072    130392    1%        4     1% /var/adm/ras/livedump
/dev/gpfs1            -         -    -         -     -  /gpfsSH1
[root on france][/] =>
```

4. Shut down the Spectrum Scale daemon on all nodes of the surviving site. Use the **mmshutdown** command to shut down the daemon. See Example 7-41.

Example 7-41 Shut down the Spectrum Scale daemon on all nodes of the surviving site

```
[root on france][/] => mmshutdown
Thu Nov 6 15:01:39 EDT 2014: mmshutdown: Starting force unmount of GPFS file
systems
forced unmount of /gpfsSH1
Thu Nov 6 15:01:44 EDT 2014: mmshutdown: Shutting down GPFS daemons
Shutting down!
'shutdown' command about to kill process 4849724
Thu Nov 6 15:01:49 EDT 2014: mmshutdown: Finished
[root on france][/] =>

[root on ghana][/] => mmshutdown
Thu Nov 6 15:02:01 EDT 2014: mmshutdown: Starting force unmount of GPFS file
systems
forced unmount of /gpfsSH1
Thu Nov 6 15:02:10 EDT 2014: mmshutdown: Shutting down GPFS daemons
Shutting down!
'shutdown' command about to kill process 4359767
```

```
Thu Nov 6 15:02:15 EDT 2014: mmshutdown: Finished
```

5. Consider changing the primary configuration server in the DR site if required in your environment before moving to the next step. This is an optional step to manage the cluster configuration using the primary server in the surviving site. In our scenario, we change the primary role to site KGN to node france and disable the existing secondary role. See Example 7-42.

Example 7-42 Assigning the primary configuration server to node france

```
[root on france][/] mmchcluster -s "" -p france
mmchcluster: GPFS cluster configuration servers:
mmchcluster:   Primary server:    france
mmchcluster:   Secondary server:  (none)
mmchcluster: Propagating the new server information to the rest of the nodes.
mexico:  mmdsh: Unexpected error from /usr/bin/scp -p
/var/mmfs/tmp/stgSdrfsFile.mmchcluster.8847614
root@mexico:/var/mmfs/tmp/stgSdrfsFile.mmchcluster.8847614. Return code: 1
mmchcluster: The following nodes are not aware of the configuration server change:
        mexico
        japan
        england
   Do not start GPFS on the above nodes until the problem is resolved.
mmchcluster: Make sure that the following nodes are available:
        france
mmchcluster: Run the mmchcluster -p LATEST command until successful.
```

6. We change node quorum for mexico and japan. To relax the node quorum requirements, change mexico, japan, and england to *nonquorum* with the **mmchnode --nonquorum** command. See Example 7-43.

Example 7-43 Change mexico, japan, and england to nonquorum nodes

```
[root on france][/] => mmchnode --nonquorum  -N mexico,japan,england
Thu Nov 6 15:55:16 EDT 2014: mmchnode: Processing node mexico
Thu Nov 6 15:55:16 EDT 2014: mmchnode: Processing node japan
Verifying GPFS is stopped on all affected nodes ...
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on france][/] =>
```

7. Change disk descriptor quorum for site-wide file system. Relax the file system descriptor quorum by using the **mmfsctl** command. See Example 7-44.

*Example 7-44 Use **mmfsctl** to relax file system descriptor quorum*

```
[root on france][/] => mmfsctl gpfs1 exclude -d "gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs7nsd"
mmfsctl: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on france][/] =>
```

8. Start IBM Spectrum Scale on all nodes at the KGN site. Use **mmstartup** to restart daemons at the surviving site. See Example 7-45.

Example 7-45 Restart the Spectrum Scale daemons at the surviving site

```
[root on france][/] => mmstartup -N france,ghana
Thu Nov 6 15:13:45 EDT 2014: mmstartup: Starting GPFS ...
[root on france][/] =>
```

9. Check the file system and disk on the surviving site; verify that gpfs1 is mounted. See Example 7-46.

Example 7-46 Verify gpfs1 is mounted on all nodes of the surviving site

```
[root on france][/] => mmlsmount all -L
File system gpfs1 is mounted on 2 nodes:
  10.11.12.123 france
  10.11.12.124 ghana
[root on france][/] =>

[root on france][/] => mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         down         system
gpfs2nsd     nsd         512    3001 yes      yes   ready         down         system
gpfs3nsd     nsd         512    3001 yes      yes   ready         down         system
gpfs4nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs5nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs6nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs7nsd     nsd         512    3003 no       no    ready         down         system
[root on france][/] =>
```

Fail back after temporary outage and configuration change

If the failure was of a temporary nature and the configuration has changed, use the steps in this section to restore the original configuration. Ensure that all nodes at the failed site are repaired and any disk problems are corrected. In our scenario, the Spectrum Scale services on the surviving site are started and the file system gpfs1 mounted when starting the recovery procedure. The recovered nodes in sites POK and BUF are up and the Spectrum Scale services are not started. We perform the following steps:

1. Ensure that all nodes contain the latest `mmsdrfs` file. Use **`mmchcluster`** to check for the most recent levels. See Example 7-47.

Example 7-47 Make sure the mmsdrfs file on each node is at the latest level

```
[root on france][/] => mmchcluster -p LATEST
mmchcluster: GPFS cluster configuration servers:
mmchcluster:   Primary server:   france
mmchcluster:   Secondary server: none
mmchcluster: Propagating the new server information to the rest of the nodes.
mmchcluster: Command successfully completed
[root on france][/]=>
```

2. Move the primary configuration server back to the POK site and the secondary cluster configuration server to the KGN site. See Example 7-48.

Example 7-48 Move the secondary cluster configuration server france back to the KGN site

```
[root on france][/] => mmchcluster -p mexico -s france
mmchcluster: GPFS cluster configuration servers:
mmchcluster:   Primary server:    mexico
mmchcluster:   Secondary server:  france
mmchcluster: Propagating the new server information to the rest of the nodes.
mmchcluster: Command successfully completed
```

```
[root on france][/] =>
```

3. Start the Spectrum Scale services on the recovered nodes: mexico, japan, and england. See Example 7-49.

Example 7-49 Starting the Spectrum Scale services on the recovered nodes

```
[root on france][/] mmstartup -N mexico,japan,england
Thu Nov  6 16:20:50 EDT 2014: mmstartup: Starting GPFS ...
```

4. Restore original node quorum designation. Use the **mmchnode** command. See Example 7-50. Observe that the node designation for POK is performed while Spectrum Scale services are running in the KGN site.

Example 7-50 Change ghana and france back to quorum nodes

```
[root on france][/] => mmchnode --quorum  -N mexico,japan,england
Thu Nov 6 16:28:24 EDT 2014: mmchnode: Processing node mexico
Thu Nov 6 16:28:24 EDT 2014: mmchnode: Processing node japan
Thu Nov 6 16:28:24 EDT 2014: mmchnode: Processing node england
mmchnode: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
[root on france][/] =>
```

5. Restore the file system descriptor quorum. Use the **mmfsctl** command to restore file system descriptor quorum. See Example 7-51.

Note: **mmfsctl include/exclude** commands require the Spectrum Scale file system to be unmounted.

Example 7-51 Use mmfsctl to restore file system descriptor quorum

```
[root on france][/] =>mmumount gpfs1 -a
Thu Nov 6 16:59:18 EDT 2014: mmumount: Unmounting file systems ...
[root on france][/] => mmfsctl gpfs1 include  -d "gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs7nsd"
mmfsctl: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. Bring the disks online by using the **mmchdisk** command and mount back the Spectrum Scale file system. See Example 7-52.

Example 7-52 Use mmchdisk to bring all “down” disks of the gpfs1 file system back online

```
[root on mexico][/] => mmchdisk gpfs1 start -d "gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs7nsd"
mmnsddiscover:  Attempting to rediscover the disks.  This may take a while ...
mmnsddiscover:  Finished.
.......
Scanning file system metadata, phase 1 ...
  59 % complete on Thu Nov 6 17:15:26 2014
 100 % complete on Thu Nov 6 17:15:28 2014
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Thu Nov 6 17:18:32 2014
```

```
Scan completed successfully.
[root on mexico][/] => mmmount gpfs1 -a
Thu Nov  6 17:18:43 EDT 2014: mmmount: Mounting file systems ...
```

7. Run the **mmrestripefs** command to restripe the file system across all disks and restore the initial replication proprieties. See Example 7-53.

Example 7-53 Restripe the file system on all cluster disks

```
[root on mexico][/] => mmrestripefs gpfs1 -b
Scanning file system metadata, phase 1 ...
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
 100.00 % complete on Thu Nov  6 17:20:32 2014  (    225408 inodes with total       1045
MB data processed)
Scan completed successfully.
```

8. Check the state of the Spectrum Scale daemon at all sites. Use the **mmgetstate** command to get the state of the daemons. See Example 7-54.

Example 7-54 Verify all Spectrum Scale daemons are started all sites

```
[root on ghana][/] => mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      mexico             3        5          5       active      quorum node
       2      japan              3        5          5       active      quorum node
       3      france             3        5          5       active      quorum node
       4      ghana              3        5          5       active      quorum node
       5      england            3        5          5       active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            5
Number of local nodes active in the cluster:       5
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     5
Number of quorum nodes active in the cluster:      5
Quorum = 3, Quorum achieved

[root on ghana][/] =>
```

9. Check mount status of gpfs1 file system at all sites. Verify that the gpfs1 file system is mounted on all nodes of all sites. See Example 7-55.

Example 7-55 Verify file system gpfs1 is mounted on all nodes of all sites

```
[root on france][/] => mmlsmount gpfs1 -L
File system gpfs1 is mounted on 5 nodes:
  10.11.12.121    mexico
  10.11.12.122    japan
  10.11.12.123    france
  10.11.12.124    ghana
```

```
  10.11.12.125    england
[root on france][/] =>
```

10. Check the state of all disks in the file system to verify their availability. See Example 7-56.

Example 7-56 Verify all disks in file system /gpfs1 are up and available

```
[root on france][/] => mmlsdisk gpfs1
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
gpfs1nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs2nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs3nsd     nsd         512    3001 yes      yes   ready         up           system
gpfs4nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs5nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs6nsd     nsd         512    3002 yes      yes   ready         up           system
gpfs7nsd     nsd         512    3003 no       no    ready         up           system
[root on france][/] =>
```

7.3 Backup and restore for IBM Spectrum Scale

This section describes the IBM Spectrum Scale backup tools that are used for protecting the Spectrum Scale file system data and also for saving file system configuration. The following sections are detailed:

- Spectrum Scale backup tools: **mmbackup**, combined with the Tivoli Storage Manager backup-archive client are the base utilities for backing up and restoring a Spectrum Scale file system.
- Spectrum Scale advanced backup tool presents the **mmbackupconfig** and **mmrestoreconfig** utilities.
- SOBAR is also an advanced mechanism for data protection against disaster only for Spectrum Scale file systems that are managed by Tivoli Storage Manager hierarchical storage management (HSM).

Note: IBM Spectrum Scale 4.1 Standard Edition, or higher is required for running the preceding backup and recovery tools.

7.3.1 mmbackup utility

IBM Spectrum Scale includes the **mmbackup** utility to perform backup of the Spectrum Scale file systems data.

The **mmbackup** command can be used to back up:

- An entire Spectrum Scale file system
- A Spectrum Scale snapshot

The **mmbackup** command relies on the Tivoli Storage Manager commands to back up the file system. You need to install and configure the Tivoli Storage Manager backup-archive client with the same version on each node performing the backup operations in order to use **mmbackup** command. The file systems must be mounted on the nodes specified in the **mmbackup** command. Follow the Tivoli Storage Manager Backup-Archive installation and

configuration manuals in order to complete the Tivoli Storage Manager installation and configuration before using **mmbackup** commands.

Consider when using the **mmbackup** that although the file system data is saved, the file system configuration cannot be saved because the Tivoli Storage Manager tools do not save specific information about the file system configuration. Use the **mmbackupconfig** command to save additional information on the file system metadata, such as: filesets, storage pool information, policy, and quota.

The **mmbackup** command has several options for processing, which are explained in the following sections. It allows for two types of backup:

- Full backup: performs a full backup of file system data, regardless of previous backups performed.
- Incremental backup: this is the default option. It performs a backup only of the changed data.

How the mmbackup command is integrated with Tivoli Storage Manager

The **mmbackup** command performs several operations in order to save the data in the Spectrum Scale file system and sync up with data already backed up:

- Determines the file selection filters, which apply for the backup session by calling the **dsmc query inclexcl** command, which reads the include/exclude statements from the client options file or via a client options set on the Tivoli Storage Manager server.
- Spectrum Scale Scan engine translates the include/exclude filters into Spectrum Scale policy rules. These rules are applied by using the **mmaplypolicy** command. The file system is then scanned for changes and compared with the existing shadow database. This database is stored in the root of the Spectrum Scale file system and it is updated by each scan of **mmbackup**. The file is named: `.mmbackupShadow.1.<tsm-server-stanza-name>`. When using multiple Tivoli Storage Manager servers, an individual file is created for each server.
- The result of the Spectrum Scale scan is passed into two types of Tivoli Storage Manager operations, avoiding a standard file system scan by Tivoli Storage Manager, required in the regular backup cases:
  - Files new/changed are passed in a list for Tivoli Storage Manager backup: **dsmc selective filelist=<filelist_backup>**. In case of files being modified only in metadata (for example: ownership, permissions, modification time), then an incremental backup is called.
  - Files deleted, which need to be deactivated on the Tivoli Storage Manager server are passed in a list to command: **dsmc expire filelist=<filelist_inactivate>**.

The mmbackup flow is illustrated in Figure 7-6 on page 375.

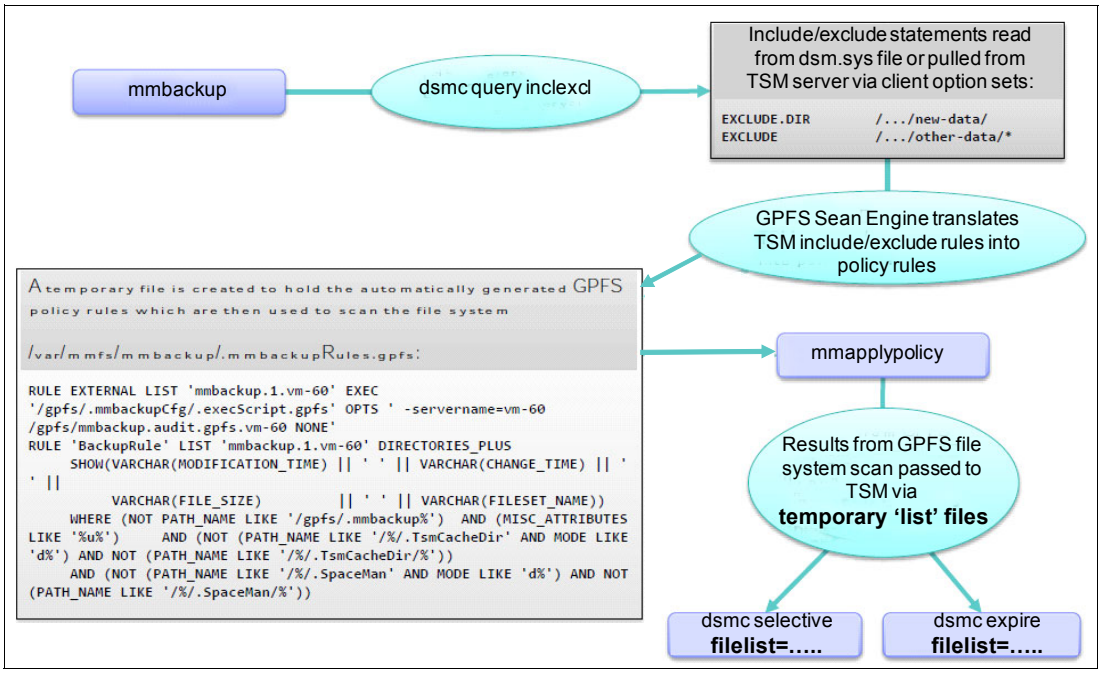

mmbackup
dsmc query inclexcl
Include/exclude statements read from dsm.sys file or pulled from TSM server via client option sets:
EXCLUDE.DIR /.../new-data/
EXCLUDE /.../other-data/*
GPFS Sean Engine translates TSM include/exclude rules into policy rules
A temporary file is created to hold the automatically generated GPFS policy rules which are then used to scan the file system
/var/mmfs/mmbackup/.mmbackupRules.gpfs:
RULE EXTERNAL LIST 'mmbackup.1.vm-60' EXEC
'/gpfs/.mmbackupCfg/.execScript.gpfs' OPTS ' -servername=vm-60
/gpfs/mmbackup.audit.gpfs.vm-60 NONE'
RULE 'BackupRule' LIST 'mmbackup.1.vm-60' DIRECTORIES_PLUS
SHOW(VARCHAR(MODIFICATION_TIME) || ' ' || VARCHAR(CHANGE_TIME) || '
' ||
VARCHAR(FILE_SIZE) || ' ' || VARCHAR(FILESET_NAME))
WHERE (NOT PATH_NAME LIKE '/gpfs/.mmbackup%') AND (MISC_ATTRIBUTES
LIKE '%u%') AND (NOT (PATH_NAME LIKE '/%/.TsmCacheDir' AND MODE LIKE
'd%') AND NOT (PATH_NAME LIKE '/%/.TsmCacheDir/%'))
AND (NOT (PATH_NAME LIKE '/%/.SpaceMan' AND MODE LIKE 'd%') AND NOT
(PATH_NAME LIKE '/%/.SpaceMan/%'))
mmapplypolicy
Results from GPFS file system scan passed to TSM via temporary 'list' files
dsmc selective filelist=.....
dsmc expire filelist=.....

Figure 7-6 mmbackup flow

Considerations for mmbackup

When using **mmbackup** for backing up the Spectrum Scale file systems, consider the following aspects:

- Tivoli Storage Manager Backup and Archive client V7.1.1 is supported for use with IBM Spectrum Scale V4.1.0.
- File systems with Tivoli Storage Management for Hierarchical Storage Manager that have unlinked filesets will be required to link all filesets when issuing the **mmbackup** command for the first time after you upgrade a cluster from GPFS 3.5.0.10 or lower, to GPFS 3.5.0.11 or higher versions.
- Use of the Tivoli Storage Manager Backup and Archive client option **SKIPACLUPDATECHECK** with the **mmbackup** command requires Tivoli Storage Manager release 6.4.1.0, or later. The **SKIPACLUPDATECHECK** option disables checksum and size comparisons of ACL data. When set to yes (default is no), the Tivoli Storage Manager client will not perform checksum and size comparisons before or after backup and during incremental processing (ACL checksum from previous backup and current ACL) to detect ACL updates.
- The Spectrum Scale **mmbackup** command is not l with the Tivoli Storage Management for Hierarchical Storage Manager managing a file system with multiple Tivoli Storage Manager servers.
- Starting with IBM Spectrum Scale 4.1, the **mmbackup** command will no longer support the path name: `/<GPFSMountPoint>/.snapshots/.mmbuSnapshot` for incremental backups. After migrating to IBM Spectrum Scale 4.1, if the older `.mmbuSnapshot` path name format is still in use, a full backup is required if a full backup has never been performed with GPFS 3.3 or later. After the full backup is performed, files will now always be stored in Tivoli Storage Manager under their usual root directory path name. All files in Tivoli Storage Manager under `/<GPFSMountPoint>/.snapshots/.mmbuSnapshot` will be marked for expiration automatically after a successful backup.
- Restoring a file via a node that has a different architecture than the one used to do the backup could cause the associated ACL to be corrupted. For example, if a file was backed up using an x86_64 node, and then restored using a ppc64 node, this could cause its ACL

to be corrupted. We recommend that backup and restore operations be done on similar types of nodes.

- When the **mmbackup** command is run on a file system that has AFM filesets, only cached data from the AFM filesets gets backed up; **mmbackup** will not back up uncached data from the AFM filesets.

Multiple node option

The **mmbackup** command can use multiple nodes to process the backup request. Because all nodes have the same list of objects for backup, the data saved needs to be owned by a single node in Tivoli Storage Manager containing the result of the backup operations on all nodes. This configuration can be achieved by using a proxy node in the Tivoli Storage Manager configuration, which is used by every Spectrum Scale node during the backup operation. Figure 7-7 illustrates the proxy node role and the relationship with the Spectrum Scale nodes. Each individual node is defined in Tivoli Storage Manager (GPFS_node1, GPFS_node2, so on) and the GPFS_proxy node, which owns all the data saved by any Spectrum Scale node in the cluster.

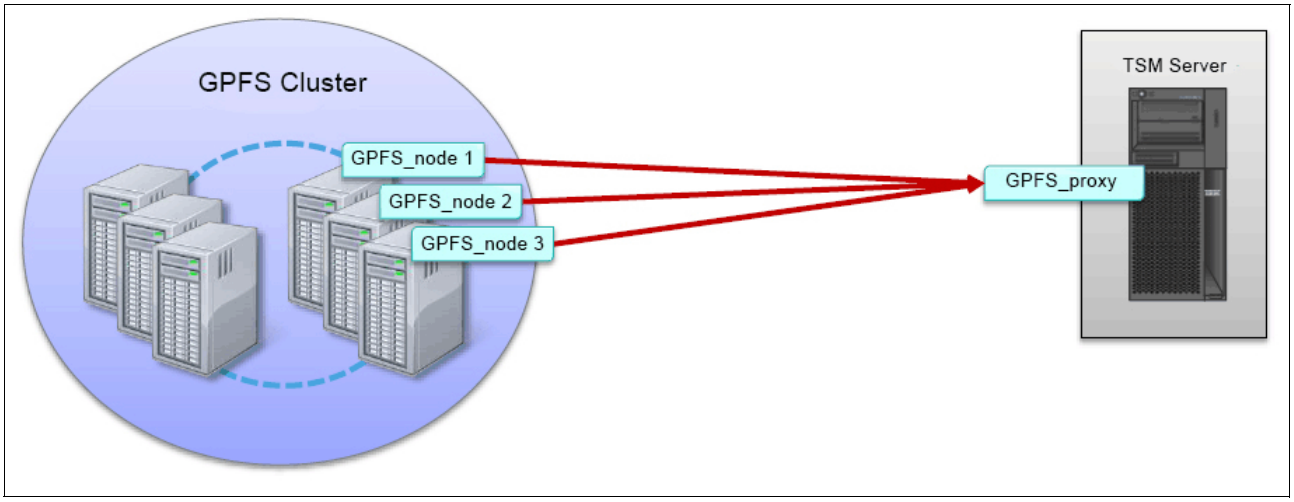


Figure 7-7 Spectrum Scale cluster and the Tivoli Storage Manager proxy node

Using multiple nodes for processing the backup request can be specified by using **-N** parameter with a list of nodes, or a file containing the list of nodes, or a node class grouping a selection of nodes. Example 7-57 presents a full backup task for the Spectrum Scale file system `/gpfs1` on nodes tsmcl1 and tsmcl2.

Example 7-57 Using multiple nodes for processing the backup request

```
root@tsmcl1:/>mmbackup /gpfs1/ -t full -N tsmcl1,tsmcl2
-------------------------------------------------------
mmbackup: Backup of /gpfs1 begins at Wed Oct 22 09:31:16 EDT 2014.
-------------------------------------------------------
Wed Oct 22 09:31:37 2014 mmbackup:Scanning file system gpfs1
Wed Oct 22 09:31:57 2014 mmbackup:Determining file system changes for gpfs1
[tsmgpfs1].
Wed Oct 22 09:31:57 2014 mmbackup:changed=18377, expired=0, unsupported=0 for
server [tsmgpfs1]
Wed Oct 22 09:31:58 2014 mmbackup:Sending files to the TSM server [18377 changed,
0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      19926
```

```
        Total number of objects backed up:      19926
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of bytes transferred:      825
------------------------------------------------------------
mmbackup: Backup of /gpfs1 completed successfully at Wed Oct 22 09:35:30 EDT 2014.
------------------------------------------------------------
```

The client sessions opened by the previous backup task can be observed in the Tivoli Storage Manager admin session by using the **query session** command, as shown in Example 7-58.

Example 7-58 Tivoli Storage Manager client sessions

```
tsm: TSMGPFS1>query session

  Sess      Comm.      Sess        Wait       Bytes       Bytes     Sess        Platform     Client Name
Number      Method     State       Time        Sent       Recvd     Type
------      ------     ------     ------     -------     -------     -------     --------     ------------------
   405      Tcp/Ip     IdleW       0 S       253.4 K     109.9 K     Node        AIX          GPFS_PROXY
                                                                                              (TSMCL1)
   407      Tcp/Ip     IdleW       0 S         1.1 K     259.7 M     Node        AIX          GPFS_PROXY
                                                                                              (TSMCL1)
   409      Tcp/Ip     Run         0 S         1.2 K         263     Admin       AIX          ADMIN
   410      Tcp/Ip     Run         0 S        15.2 K       6.7 K     Node        AIX          GPFS_PROXY
                                                                                              (TSMCL2)
   411      Tcp/Ip     Run         0 S          780      31.8 M     Node        AIX          GPFS_PROXY
                                                                                              (TSMCL2)
```

Using filesets

When using filesets inside the Spectrum Scale file system, the **mmbackup** command verifies if the filesets are linked, otherwise the command execution stops. You can specify the **-f** flag in order to ignore the unlinked filesets only with incremental operations, like in the following example:

```
mmbackup gpfs -t incremental -f
```

Note: Using **-f** determines the file included in the linked fileset to be skipped from backup, not expired. It can have a large impact on performance. We recommend to avoid using it unless performing a backup operation with unlinked filesets is necessary.

Rebuilding the shadow database

The mmbackup shadow database can be rebuilt based on the information in the Tivoli Storage Manager server, by using the **--rebuild** option. See Example 7-59. A rebuild operation will synchronize the shadow database information with the actual information in Tivoli Storage Manager for all file system objects.

Example 7-59 Rebuilding the shadow database

```
root@tsmcl1:/>mmbackup gpfs1 --rebuild
------------------------------------------------------------
mmbackup: Shadow database rebuild of /gpfs1 begins at Wed Oct 22 11:50:27 EDT
2014.
------------------------------------------------------------
Wed Oct 22 11:50:52 2014 mmbackup:Querying files currently backed up in TSM
server:tsmgpfs1.
```

```
Wed Oct 22 11:51:01 2014 mmbackup:Built query data file from TSM server: tsmgpfs1
rc = 0
Wed Oct 22 11:51:09 2014 mmbackup:Scanning file system gpfs1
Wed Oct 22 11:51:13 2014 mmbackup:Reconstructing previous shadow file
/gpfs1/.mmbackupShadow.1.tsmgpfs1 from query data for tsmgpfs1
Wed Oct 22 11:51:14 2014 mmbackup:Done with shadow file database rebuilds
-------------------------------------------------------
mmbackup: Shadow database rebuild of /gpfs1 completed successfully at Wed Oct 22
11:51:14 EDT 2014.
-------------------------------------------------------
```

Another option for rebuilding the shadow database file is using command: **mmbackup <gpfs_filesystem> -q**. In this case, the shadow file will be rebuilt as in the previous case, then a file system backup operation is started.

Note: The rebuild operation is a resource-intensive operation (CPU and memory). It exchanges with the Tivoli Storage Manager server information about all objects on the file system and it might take a significant amount of time, particularly if the Spectrum Scale file systems have a very large number of files.

Backing up of Spectrum Scale snapshots

You can use this feature for saving the Spectrum Scale snapshots. This can be useful for backing up point-in-time copies of the file system, previously generated in the Spectrum Scale file system.

Note: Backup of snapshots requires GPFS 3.5.0.3, or later versions.

The **mmbackup** command excludes by default the snapshot data in a Spectrum Scale file system. You can back up the snapshots for a file system by using the **-S** option, like in Example 7-60.

Example 7-60 Backup of Spectrum Scale snapshot

```
root@tsmcl1:/gpfs1>mmbackup gpfs1 -S snap.2
-------------------------------------------------------
mmbackup: Backup of /gpfs1 begins at Wed Oct 22 14:54:31 EDT 2014.
-------------------------------------------------------
Wed Oct 22 14:55:04 2014 mmbackup:Scanning file system gpfs1
Wed Oct 22 14:55:08 2014 mmbackup:Determining file system changes for gpfs1
[tsmgpfs1].
Wed Oct 22 14:55:09 2014 mmbackup:changed=1, expired=0, unsupported=0 for server
[tsmgpfs1]
Wed Oct 22 14:55:09 2014 mmbackup:Sending files to the TSM server [1 changed, 0
expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      1
        Total number of objects backed up:      1
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of bytes transferred:      2
-------------------------------------------------------
```

```
mmbackup: Backup of /gpfs1 completed successfully at Wed Oct 22 14:55:30 EDT 2014.
-----------------------------------------------------------
```

Note: When backing up incrementally the snapshots with **mmbackup**, the changed data is evaluated based on the shadow file, as in the regular file system backup. If an incremental backup is run on a snapshot after a file system backup, the applicable changes for the snapshot image are determined comparing with the last backup operation run on the file system. There is no dedicated file space in Tivoli Storage Manager for the snapshots, and there are no new objects saved in Tivoli Storage Manager resulted from the specific access location of the snapshot inside the Spectrum Scale file system.

Using multiple Tivoli Storage Manager servers

The **mmbackup** command provides the capability to save the data in multiple Tivoli Storage Manager servers when performing a backup operation. The **mmbackup** can be used for specifying different Tivoli Storage Manager servers for backing up different Spectrum Scale file systems or for backing up the same file system to different Tivoli Storage Manager servers. In the last case, the file system data is partitioned into multiple Tivoli Storage Manager servers, and not saved multiple times in different locations. This can be used with file systems having a very large number of files. Consider when backing up file systems with a large number of files the Tivoli Storage Manager database capacity required to store the backup metadata information. Table 7-4 shows the maximum number of objects that are supported by various Tivoli Storage Manager versions.

Table 7-4 Tivoli Storage Manager versions and database limitations

Tivoli Storage Manager server version	Maximum DB size	Max. no. of objects
6.1	1 TB	1 Billion
6.2	2 TB	2 Billion
6.3	4 TB	4 Billion
7.1	4 TB	4 Billion

The server stanza must be present in the client option file (`dsm.sys`) and the server name can be specified in the **mmbackup** command, like in the following example:

```
mmbackup gpfs –t incremental --tsm-servers tsmsrv1,tsmsrv2
```

Spectrum Scale file system objects are grouped together in increments of approximately 100 and distributed round robin to each of the Tivoli Storage Manager servers and Spectrum Scale nodes specified. The **-B** flag used with **mmbackup** governs the size of the object groupings. We recommended modifying the default value from 100 to a much higher number if the file system being backed up has many small objects. For example, to use the grouping factor of 1000 files, use the following command:

```
mmbackup gpfs –t incremental --tsmsrv1,tsmsrv2 –B 1000
```

Tuning backups using mmbackup (new in IBM Spectrum Scale 4.1)

The **mmbackup** command has several parameters that can be used to tune the backup jobs. The tuning options for **mmbackup** require IBM Spectrum Scale 4.1, or later. These parameters need to be tuned according to each environment's characteristics. The parameters apply to all execution phases of the **mmbackup** command:

- During the scanning phase, the resources that **mmbackup** uses on each node specified with the **-N** parameter can be controlled:
  - The **-a IscanThreads** parameter allows specification of the number of threads and sort pipelines each node will run during the parallel inode scan and policy evaluation. This parameter affects the execution of the high-performance protocol that is used when both the **-g** and **-N** parameters are specified. The default value is 2. Using a moderately larger number can significantly improve performance, but might strain the resources of the node. In some environments, a large value for this parameter can lead to a command failure.

    Note: Set this parameter to the number of CPU cores that are implemented on a typical node in your Spectrum Scale cluster.

  - The **-n DirThreadLevel** parameter allows specification of the number of threads that are created and dispatched within each mmapplypolicy process during the directory scan phase. The default value is 24.
- During the execution phase for expire, mmbackup processing can be adjusted as follows:
  - Automatic computation of the ideal expire bunch count. The number of objects named in each file list can be determined, separately from the number in a backup list, and automatically computed if not specified by the user.
  - As an alternative to the automatic computation, the user can control expire processing as follows:
    - The **--max-expire-count** parameter can be used to specify a bunch-count limit for each **dsmc expire** command. The default value is 4. This parameter cannot be used with **-B** (max files option).
    - The **--expire-threads** parameter can be used to control how many threads run on each node running dsmc expire. This parameter cannot be used with **-m**. The default value is 4.
- During the execution phase for backup, mmbackup processing can be adjusted as follows:
  - Automatic computation of ideal backup bunch count. The number of objects named in each file list can be determined, separately from the number in an expire list, and automatically computed if not specified by the user.
  - As an alternative to the automatic computation, the user can control backup processing as follows:
    - The **--max-backup-count** parameter can be used to specify a bunch-count limit for each dsmc selective or dsmc incremental command. This parameter cannot be used with **-B** (max files option).
    - The **--backup-threads** parameter can be used to control how many threads run on each node running backup. This parameter cannot be used with **-m**. It defaults to 1 (single thread backup).
    - The **--max-backup-size** parameter can be used to further limit the size of a backup bunch by the overall size of all files that are listed in any single bunch list.

Restoring Spectrum Scale data

The data saved with **mmbackup** or saved using the Tivoli Storage Manager Backup-Archive client can be restored to the Spectrum Scale file system by using the TSM Backup-Archive client (as an example, **dsmc** command).

There can be several choices for data restore:

- Partial file system restore: You can restore a file, directory, or a group of files using wildcards. In Example 7-61, we restore the `/gpfs1/TSMCLI_AIX` subdirectory previously saved in Tivoli Storage Manager by using the **mmbackup** command.

Example 7-61 Restoring a directory from Tivoli Storage Manager backup

```
root@tsmcl2:/gpfs1>dsmc restore -subdir=yes /gpfs1/TSMCLI_AIX/
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 0.0
  Client date/time: 10/22/14   16:52:59
(c) Copyright by IBM Corporation and other(s) 1990, 2013. All Rights Reserved.

Node Name: TSMCL2
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 10/22/14   16:52:59  Last access: 10/22/14   16:52:10

Accessing as node: GPFS_PROXY
Restore function invoked.

ANS1247I Waiting for files from the server...
Restoring           4,096 /gpfs1/TSMCLI_AIX [Done]
Restoring           4,096 /gpfs1/TSMCLI_AIX/usr [Done]

...........

Restore processing finished.

Total number of objects restored:            29
Total number of objects failed:               0
Total number of bytes transferred:       271.78 MB
Data transfer time:                        1.69 sec
Network data transfer rate:          163,991.19 KB/sec
Aggregate data transfer rate:         39,060.04 KB/sec
Elapsed processing time:               00:00:07
```

- Full file system restore. An entire file system restore can be performed in a similar manner by providing the Spectrum Scale file system mount point path in the **dsmc restore** command, like in the following example:

```
dsmc restore -subdir=yes /gpfs1/
```

Note: The Spectrum Scale file system needs to be created and mounted before restoring the data from Tivoli Storage Manager. The Tivoli Storage Manager client restore process does not restore the file system configuration, such as pools, or filesets. They can be saved by using the **mmbackupconfig** command. See 7.3.2, “Spectrum Scale advanced backup tools” on page 383.

Tivoli Storage Manager restore operations and Spectrum Scale extended attributes

Tivoli Storage Manager can back up and restore the extended attributes of the files in a Spectrum Scale file system. The attributes of a file or directory can be listed by using the

mmlsattr command. In Example 7-62, we list the attributes of a file that are placed in the storage pool “silver” and is also part of the fileset “tstfset”.

Example 7-62 Listing Spectrum Scale attributes of a file

```
root@tsmcl2:/gpfs1/tstfset> mmlsattr -L file1.silver
file name:            file1.silver
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    silver
fileset name:         tstfset
snapshot name:
creation time:        Wed Oct 22 17:40:09 2014
Windows attributes:   ARCHIVE
Encrypted:            no
```

When restoring the files and directory objects of the Spectrum Scale file system, the Tivoli Storage Manager client automatically attempts to restore the Spectrum Scale attributes of the object. In our example, a restore of the `file1.silver` file is performed to the same Spectrum Scale pool and fileset. See a detailed output in Example 7-63.

Example 7-63 Restoring a file from Tivoli Storage Manager with special attributes

```
root@tsmcl2:/>rm /gpfs1/tstfset/file1.silver
root@tsmcl2:/>ls -al /gpfs1/tstfset/file1.silver
/gpfs1/tstfset/file1.silver not found
root@tsmcl2:/>dsmc restore /gpfs1/tstfset/file1.silver
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 0.0
  Client date/time: 10/22/14   17:57:13
(c) Copyright by IBM Corporation and other(s) 1990, 2013. All Rights Reserved.

Node Name: TSMCL2
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 10/22/14   17:57:13  Last access: 10/22/14   17:56:12

Accessing as node: GPFS_PROXY
Restore function invoked.

Restoring     104,857,600 /gpfs1/tstfset/file1.silver [Done]

Restore processing finished.

Total number of objects restored:             1
Total number of objects failed:               0
Total number of bytes transferred:       100.00 MB
Data transfer time:                        0.91 sec
Network data transfer rate:          111,498.78 KB/sec
Aggregate data transfer rate:         32,896.63 KB/sec
Elapsed processing time:               00:00:03

root@tsmcl2:/> mmlsattr -L /gpfs1/tstfset/file1.silver
file name:            /gpfs1/tstfset/file1.silver
```

```
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    silver
fileset name:         tstfset
snapshot name:
creation time:        Wed Oct 22 17:57:20 2014
Windows attributes:   ARCHIVE
Encrypted:            no
root@tsmcl2:/>
```

In case the fileset no longer exists or is not linked, the Tivoli Storage Manager restore process associates the file with the original storage pool and the default restore path to the original paths of the linked fileset, but the path is associated with the upper level fileset. It does not restore or configure the fileset.

In case of the storage pool, the selection of the destination pool is determined by the following sequence of rules:

- Match the file attributes to the current Spectrum Scale restore policy rule.
- Use the same Spectrum Scale storage pool that the file was in when it was backed up.
- Use the current Spectrum Scale file placement policy (the same as in the case of new files).
- Use the default Spectrum Scale storage pool (“system”).

7.3.2 Spectrum Scale advanced backup tools

In order to re-create the entire Spectrum Scale file system configuration, IBM Spectrum Scale provides the **mmbackupconfig** and **mmrestoreconfig** tools for backing up the configuration data, respectively restoring it.

The **mmbackupconfig** command saves the following file information:

- Block size
- Replication factors
- Number and size of disks
- Storage pool layout
- File sets and junction points
- Policy rules
- Quota information
- A number of other file system attributes

The following scenario provides the steps for backing up and recovery of an entire Spectrum Scale file system named `gpfs1` based on the **mmbackupconfig** and **mmbackup** tools.

The initial configuration of the file system contains a second storage pool “silver” and a fileset “tstfset”. Example 7-64 shows the initial configuration of the pools and filesets on `gpfs1` file system.

Example 7-64 Initial configuration of pools and filesets of gpfs1 file system

```
root@tsmcl1:/>mmlspool gpfs1
Storage pools in file system at '/gpfs1':
Name                    Id   BlkSize Data Meta Total Data in (KB)   Free Data in (KB)   Total Meta in (KB)    Free Meta
in (KB)
```

```
system                  0    512 KB  yes  yes       10485760        3108864 ( 30%)       10485760        3173376 ( 30%)
silver              65537    512 KB  yes   no      104857600      104688128 (100%)              0              0 (  0%)

root@tsmcl1:/>mmlsfileset gpfs1 -L
Filesets in file system 'gpfs1':
Name                            Id      RootInode  ParentId Created                     InodeSpace      MaxInodes    AllocInodes Comment
root                             0              3        -- Fri Oct 17 17:57:25 2014       0               66048          66048 root
fileset
tstfset                          1          60909         0 Wed Oct 22 09:54:42 2014       0                   0              0
```

For test purposes, we use a `/gpfs1/tstfset/file1.silver` file. See the attributes of the file in Example 7-65.

Example 7-65 Attributes of the test file

```
root@tsmcl1:/gpfs1/tstfset> mmlsattr -L file1.silver
file name:            file1.silver
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    silver
fileset name:         tstfset
snapshot name:
creation time:        Wed Oct 22 17:40:09 2014
Windows attributes:   ARCHIVE
Encrypted:            no
```

The following steps are applied for the recovery of the entire Spectrum Scale file system `gpfs1`:

- Back up of the Spectrum Scale file system

  a. Back up the Spectrum Scale file system `gpfs1` by using the **`mmbackup`** command. See Example 7-66.

  Example 7-66 Backup gpfs1 file system

```
root@tsmcl1:/> mmbackup gpfs1gpfs1 -t full -N tsmcl1,tsmcl2
-------------------------------------------------------
mmbackup: Backup of /gpfs1 begins at Thu Oct 23 17:10:59 EDT 2014.
-------------------------------------------------------
Thu Oct 23 17:11:36 2014 mmbackup:Scanning file system gpfs1
Thu Oct 23 17:11:57 2014 mmbackup:Determining file system changes for gpfs1
[tsmgpfs1].
Thu Oct 23 17:11:58 2014 mmbackup:changed=18381, expired=0, unsupported=0
for server [tsmgpfs1]
Thu Oct 23 17:11:58 2014 mmbackup:Sending files to the TSM server [18381
changed, 0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      18381
        Total number of objects backed up:      18381
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
```

```
        Total number of objects failed:         0
        Total number of bytes transferred:      637
-------------------------------------------------------
mmbackup: Backup of /gpfs1 completed successfully at Thu Oct 23 17:15:40 EDT
2014.
-------------------------------------------------------
```

b. Back up the file system configuration information. See Example 7-67.

Example 7-67 Backup file system configuration information

```
root@tsmcl1:/> mmbackupconfig gpfs1 -o /tmp/gpfs1.cfg

mmbackupconfig: Processing file system gpfs1 ...
mmbackupconfig: Command successfully completed
```

- Delete the file system from the Spectrum Scale cluster. First, unmount the file system using **mmunmount** and then remove the file system using **mmdelfs**. See the output in Example 7-68.

Example 7-68 Removing the file system from the Spectrum Scale cluster

```
root@tsmcl1:/>mmunmount gpfs1 -a
Thu Oct 23 17:25:26 EDT 2014: mmunmount: Unmounting file systems ...
root@tsmcl1:/>mmdelfs gpfs1
All data on the following disks of gpfs1 will be destroyed:
    gpfs4nsd
    gpfs5nsd
Completed deletion of file system /dev/gpfs1.
mmdelfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

- Recover the entire file system configuration and data

a. You need to create an empty file system containing the original pool configuration. You have the option to create the new file system with different parameters (like using a different block size) or use the original configuration from the configuration file, in our case `/tmp/gpfs1.cfg`. In our example, we re-create the original configuration of the file system `gpfs1`. The following input file is used for the output of Example 7-69:

```
%pool: pool=silver blocksize=512K
%nsd: nsd=gpfs4nsd usage=dataOnly pool=silver
%nsd: nsd=gpfs5nsd usage=dataAndMetadata pool=system
```

Example 7-69 Creating the new file system gpfs1

```
root@tsmcl1:/gpfs>mmcrfs gpfs1 -F ./gpfs1.nsd -B 512K

The following disks of gpfs1 will be formatted on node tsmcl1:
    gpfs4nsd: size 102400 MB
    gpfs5nsd: size 10240 MB
Formatting file system ...
Disks up to size 203 GB can be added to storage pool system.
Disks up to size 906 GB can be added to storage pool silver.
Creating Inode File
Creating Allocation Maps
```

```
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Formatting Allocation Map for storage pool silver
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Refer to Appendix A, “Recovery of the IBM Spectrum Scale file system configuration” on page 515 for how you can generate the scripts for creating the original file system configuration based on **mmbackupconfig** information.

b. Restore the file system configuration using **mmrestoreconfig**. See Example 7-70. This step does not require the file system to be mounted.

Example 7-70 Restore the file system attributes from backup

```
root@tsmcl1:/>mmrestoreconfig gpfs1 -i /tmp/gpfs1.cfg
------------------------------------------------------
Configuration restore of gpfs1 begins at Thu Oct 23 22:06:16 EDT 2014.
------------------------------------------------------
mmrestoreconfig: Checking disk settings for gpfs1:
mmrestoreconfig: Checking the number of storage pools defined for gpfs1.
mmrestoreconfig: Checking storage pool names defined for gpfs1.
mmrestoreconfig: Checking storage pool size for 'silver'.
mmrestoreconfig: Checking storage pool size for 'system'.

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:
File system attribute to be restored: defaultMountPoint
   Backup value:  /gpfs1
   Current value: /gpfs/gpfs1
mmchfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

mmrestoreconfig: Checking fileset configurations for gpfs1:
Fileset tstfset created with id 1 root inode 93696.

mmrestoreconfig: Mounting file system gpfs1 to link filesets.
Fileset tstfset linked at /gpfs1/tstfset
mmrestoreconfig: Fileset link status has been restored.  Unmounting file
system gpfs1.
Thu Oct 23 22:07:31 EDT 2014: mmumount: Unmounting file systems ...

mmrestoreconfig: Checking policy rule configuration for gpfs1:
mmrestoreconfig: No policy rules installed in backed up filesystem gpfs1.

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:
mmrestoreconfig: Command successfully completed
```

c. Mount the file system on all cluster nodes: *mmmount gpfs1 -a*.

d. Restore the entire file system data from backup by using TSM Backup-Archive Client (dsmc). See Example 7-71 on page 387.

Example 7-71 Restoring the GPFS file system data

```
root@tsmcl1:/gpfs>dsmc restore -subdir=yes /gpfs1/
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 0.0
  Client date/time: 10/24/14   15:42:52
(c) Copyright by IBM Corporation and other(s) 1990, 2013. All Rights
Reserved.

Node Name: TSMCL1
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 10/24/14   15:42:52  Last access: 10/24/14   15:31:50

Accessing as node: GPFS_PROXY
Restore function invoked.

ANS1247I Waiting for files from the server...
Restoring           4,096 /gpfs1/tstfset [Done]
Restoring         262,144 /gpfs1/ [Done]
Restoring     104,857,600 /gpfs1/tstfset/file1.silver [Done]
..............

Restore processing finished.

Total number of objects restored:            14
Total number of objects failed:               0
Total number of bytes transferred:         9.45 GB
LanFree data bytes:                        9.45 GB
Data transfer time:                      386.81 sec
Network data transfer rate:           25,635.28 KB/sec
Aggregate data transfer rate:         39,442.53 KB/sec
Elapsed processing time:                 00:04:11
```

e. Check the attributes of the test file `file1.silver` restored from a previous backup operation. Observe that the file has been restored with the original attributes. See Example 7-72.

Example 7-72 Checking the attributes of the restore file

```
root@tsmcl2:/gpfs1/tstfset> mmlsattr -L file1.silver
file name:            file1.silver
metadata replication: 1 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    silver
fileset name:         tstfset
snapshot name:
creation time:        Wed Oct 22 17:40:09 2014
Windows attributes:   ARCHIVE
Encrypted:            no
```

7.3.3 Scale Out Backup and Restore

Scale Out Backup and Restore (SOBAR) is a specialized mechanism for data protection against disaster only for Spectrum Scale file systems that are managed by Tivoli Storage Manager HSM.

For such systems, the opportunity exists to premigrate all file data into the HSM storage, take a snapshot of the file system structural metadata, and save a backup image of the file system structure. This metadata image backup, consisting of several image files, can be safely stored in the backup pool of the Tivoli Storage Manager server and later used to restore the file system if there is a disaster.

SOBAR uses the following commands in order to perform the backup and restore functions:

- During the backup flow: **`mmbackupconfig`** and **`mmimgbackup`**
- During the restore flow: **`mmrestoreconfig`** and **`mmimgrestore`**

These mechanisms are illustrated in Figure 7-8.

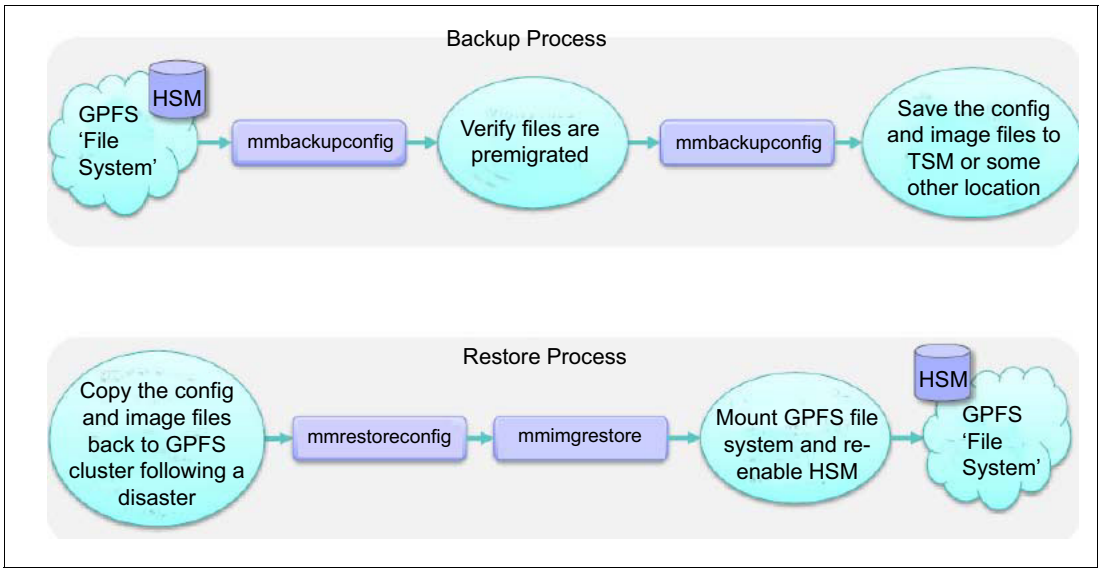


Figure 7-8 SOBAR backup/restore flows

The **`mmbackupconfig`** and **`mmrestoreconfig`** are used to save and restore the file system configuration information in this case. The **`mmimgbackup`** is used to back up the file system metadata image on the file system enabled with HSM migration. Data needs to be premigrated to Tivoli Storage Manager in order to have a complete protection of the file system data, then the **`mmimgbackup`** saves the file system metadata image. As an option, the metadata image can be saved from a snapshot, created after data is premigrated.

Note: The **`mmimgbackup`** command is designed to work with file systems using external pools, where HSM migration is enabled. Although the command can be run on a regular Spectrum Scale file system, to save a metadata image, the data backup is not covered. For regular file systems, use the **`mmbackup`** command, which automatically saves the contents of the file system. The Tivoli Storage Manager file backup client can also be used as a secondary mechanism for protecting the data from HSM-enabled file systems.

The **mmimgbackup** command has the following options (IBM Spectrum Scale V4.1 output):

```
Usage:
  mmimgbackup Device [-g GlobalWorkDirectory]
            [-L n] [-N {Node[,Node...] | NodeFile | NodeClass}]
            [-S SnapshotName] [--image ImageSetName]  [--notsm | --tsm]
            [--tsm-server servername] [POLICY-OPTIONS]
```

Following are several options that can be used with the **mmingbackup** command:

- -g GlobalWorkDirectory: The directory to be used for temporary files that need to be shared between the mmimgbackup worker nodes and to hold backup images until sent to archive. The default is: <FS_mountpoint>/.mmimgbackup.
- -L n: Controls the level of information displayed by the **mmimgbackup** command (0-6).
- -N options: Specifies the nodes participating in the backup process.
- -S SnapshotName: Specifies that the metadata image is taken from the snapshot, rather than using the file system.
- --image ImageSetName: Saves the image files using the provided argument as the base set name. Image file names use the following format:
  - ImageSetName_YYYYMMDD_hh.mm.ss_BBB.sbr, or
  - ImageSetName_YYYYMMDD_hh.mm.ss.idx
- --notsm | --tsm: Does not save the image data to Tivoli Storage Manager (only local), or saves it to Tivoli Storage Manager. You can use the '--tsm' option with '--tsm_server' to specify a particular Tivoli Storage Manager server as target. The servername is the name of the stanza in the system option file (`dsm.sys`).
- POLICY-OPTIONS: Additional options that can be specified for **mmapplypolicy** during processing (for example, number of threads during parallel inode scan, no. of scans dispatched within each mmimgbackup process during the directory scan phase, and others).

General consideration for SOBAR

Requirements and considerations for SOBAR:

- SOBAR is used in environments with IBM Spectrum Scale and ILM based on Tivoli Storage Manager HSM, and it is currently supported on AIX and Linux x86 platforms.
- SOBAR requires GPFS 3.5.0.11, or later versions used with Tivoli Storage Management for hierarchical storage manager (TSM/HSM) and Tivoli Backup-Archive client V6.4.1, or later. See more details about Tivoli Storage Manager for Space Management All Requirements web resource:

  http://www-01.ibm.com/support/docview.wss?uid=swg21321200
- SOBAR supports backup and restore of Spectrum Scale file systems at a release of 3.3.0.2 or higher.
- SOBAR does not support the backup or restore of a file system with Active File Management (AFM) filesets.
- SOBAR supports backup from a global snapshot. Independent fileset snapshots are not supported.

Backup and recovery with SOBAR

The following section describes backup considerations when recovering the Spectrum Scale cluster configuration, and the file system backup and recovery procedures with SOBAR.

General backup information for recovery of the Spectrum Scale cluster

When planning for complete recovery of the Spectrum Scale cluster, consider the following information:

- Back up the cluster configuration information. The cluster configuration must be backed up by the administrator. The minimum cluster configuration information needed is: IP addresses, node names, roles, quorum and server roles, cluster-wide configuration settings from mmchconfig, cluster manager node roles, remote shell configuration, mutual ssh and rsh authentication setup, and the cluster UID. You can use the following commands to get this information (except SSH/RSH OS configuration, which can be rebuilt):
  - **`mmlscluster`**
  - **`mmlsconfig`**

  More complete configuration information can be found in the `mmsdrfs` file.
- Preserve disk configuration information. Disk configuration must also be preserved in order to recover a file system. The basic disk configuration information needed for a backup intended for disaster recovery is the number of disk volumes that were previously available and the sizes of those volumes.

  In order to recover from a complete file system loss, at least as much disk space as was previously available is needed for restoration. It is only feasible to restore the image of a file system onto replacement disks if the disk volumes available are of similar enough sizes to the originals that all data can be restored to the new disks. At a minimum, the following disk configuration information is needed (also providing some hints on gathering it):
  - Disk device names: **`mmlsnsd -m`** shows the local device associated to each NSD. **`mmlsnsd -M`** shows all devices from all nodes associated with each NSD
  - Disk device sizes: **`mmdf`** can provide information about NSD/disk size if disk is part of the file system. You can also use OS-specific commands to get this information based on device name
  - The number of disk volumes: **`mmlsnsd`**
  - NSD server configuration: **`mmlsnsd`**
  - Disk RAID configurations: OS and storage-related tools
  - Failure group designations: **`mmdf`** and **`mmlsdisk`** show failure group for the NSD associated with a file system
  - The mmsdrfs file contents

This information should be backed up after the initial setup and each time configuration changes occur.

Tip: You can use **`mmsdrbackup`** user exit to trigger a backup operation for the `mmsdrfs` file. See more details at the following site:

`http://ibm.co/1HQ4pNP`

Sample scenario for SOBAR backup/restore

This section provides an example for using the SOBAR procedure to back up and restore a Spectrum Scale file system enabled with HSM. Our scenario consists of a cluster with two nodes: tsmcl1 and tsmcl2, connected to a Tivoli Storage Manager server (tsmsrv) for backup services. The basic setup is shown in the output of the **`mmlscluster`** and **`mmlsconfig`** commands from Example 7-73 on page 391.

Example 7-73 Spectrum Scale cluster configuration

```
root@tsmcl1:/var/mmfs/gen>mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         hsmgpfs.tsmcl1
  GPFS cluster id:           12742445374757761618
  GPFS UID domain:           hsmgpfs.tsmcl1
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1 tsmcl1            172.16.20.154  tsmcl1           quorum-manager
   2 tsmcl2            172.16.20.155  tsmcl2           quorum-manager

root@tsmcl1:/var/mmfs/gen>mmlsconfig
Configuration data for cluster hsmgpfs.tsmcl1:
----------------------------------------------
clusterName hsmgpfs.tsmcl1
clusterId 12742445374757761618
autoload yes
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
ccrEnabled yes
tiebreakerDisks tsmdisk1
prefetchThreads 100
pagepool 1500M
adminMode central

File systems in cluster tsmgpfs.tsmsrv:
---------------------------------------
/dev/gpfs1
```

We use in our scenario a file system `gpfs1` enabled for HSM, with the following disks and pools:

- System pool:
  - 2 x 10 GB LUNs for metadataOnly, allocated on SSD drives
  - 2 x 100 GB LUNs for dataOnly, allocated on SAS drives

  See the NSD capacity and usage in the **mmdf** output of Example 7-74.

Example 7-74 mmdf output for gpfs1 file system

```
root@tsmcl1:/>mmdf gpfs1
disk                disk size  failure holds    holds            free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 1.0 TB)
gpfs10nsd          104857600       -1 no       yes        99273728 ( 95%)          376 ( 0%)
gpfs9nsd           104857600       -1 no       yes        90581760 ( 86%)         1504 ( 0%)
gpfs7nsd            10485760       -1 yes      no          9881344 ( 94%)          984 ( 0%)
gpfs11nsd           10485760       -1 yes      no          9880576 ( 94%)         1224 ( 0%)
               -------------                         -------------------- -------------------
```

```
(pool total)       230686720                      209617408 ( 91%)         4088 ( 0%)

                ============                   ================== ==================
(data)             209715200                      189855488 ( 91%)         1880 ( 0%)
(metadata)          20971520                       19761920 ( 94%)         2208 ( 0%)
                ============                   ================== ==================
(total)            230686720                      209617408 ( 91%)         4088 ( 0%)

Inode Information
-----------------
Number of used inodes:           4126
Number of free inodes:         221282
Number of allocated inodes:    225408
Maximum number of inodes:      225408
```

- 'hsm' pool, defined as external pool for file system `gpfs1`. The following policy is defined for `gpfs1` file system (see Example 7-75).

Example 7-75 Active policy rules defined for gpfs1 file system

```
root@tsmcl1:/>mmlspolicy gpfs1 -L
/* Define hsm storage manager as an external pool */
RULE EXTERNAL POOL 'hsm' EXEC '/var/mmfs/etc/mmpolicyExec-hsm.sample' OPTS '-v'
/* Move data  */
RULE 'MigrateData' MIGRATE
          FROM POOL 'system'
                   THRESHOLD(25,15,10)
          TO POOL 'hsm'
          WHERE FILE_SIZE > 1024
/*Default placement rule */
RULE 'DEFAULT' SET POOL 'system'
```

We also defined a callback for applying the previous set of rules by **mmapply** when a space threshold condition occurs. The callback settings are listed in Example 7-76.

Example 7-76 Migration callback defined

```
root@tsmcl1:/>mmlscallback
....
MIGRATION
        command       = /usr/lpp/mmfs/bin/mmstartpolicy
        event         = lowDiskSpace,noDiskSpace
        parms         = %eventName %fsName
```

Connection to the Tivoli Storage Manager server is established for both nodes from our cluster: tsmcl1 and tsmcl2. The connection stanza from the `dsm.sys` file for the HSM clients is shown in Example 7-77. Observe in the output example, the proxy node name (GPFS_PROXY) used for connection to the Tivoli Storage Manager server, which runs on behalf of any of the GPFS HSM nodes in the cluster (tsmcl1, tsmcl2).

Example 7-77 Connectivity stanza for tsmcl1 and tsmcl2

```
SErvername  tsmgpfs1
   COMMMethod         TCPip
   TCPPort            1500
   TCPServeraddress   172.16.20.153
   PASSwordaccess       generate
```

```
ASNODENAME    GPFS_PROXY
Enablelanfree         yes
LANFREECommmethod     sharedmem
```

Backing up the file system with SOBAR

In order to perform the backup of the Spectrum Scale cluster information and of the file system `gpfs1`, perform the following steps:

1. Back up general cluster information in Tivoli Storage Manager:
   - In our scenario, we configured the mmsdrbackup user exit by copying the sample file `/usr/lpp/mmfs/samples/mmsdrbackup.sample` to directory `/var/mmfs/etc/mmsdrbackup` on one of the cluster nodes, and edit it as shown in Example 7-78.

   Example 7-78 mmsdrbackup script for automatic save of the mmsdrfs file

   ```
   root@tsmcl1:/var/mmfs/etc>cat mmsdrbackup
   #!/bin/ksh
   version=$1  # version number of file /var/mmfs/gen/mmsdrfs
   cp  /var/mmfs/gen/mmsdrfs /backup/mmsdrfs
   dsmc sel /backup/mmsdrfs

   return 0
   ```

   We copy the **mmsdrbackup** script to all cluster nodes in directory: `/var/mmfs/etc`.

   Note: The `mmsdrfs` file is saved to local directory `/backup` on the node performing the configuration change and also to Tivoli Storage Manager. The script can also be run manually to back up the `mmsdrfs` file. Versions of this file can be kept in Tivoli Storage Manager by using the Tivoli Storage Manager policy settings.

   - We save the NSD and disk information for the file system `gpfs1` in a local directory `/backup` on a Spectrum Scale node, and also in Tivoli Storage Manager using the proxy nodename gpfs_proxy. See Example 7-79.

   Example 7-79 Saving NSD information to Tivoli Storage Manager

   ```
   root@tsmcl1:/>mmlsnsd > /backup/mmlsnsd
   root@tsmcl1:/>mmlsnsd -M > /backup/mmlsnsd_M
   root@tsmcl1:/>mmdf gpfs1 > /backup/mmdf.gpfs1
   root@tsmcl1:/>dsmc incr -sub=yes /backup/
   IBM Tivoli Storage Manager
   Command Line Backup-Archive Client Interface
     Client Version 7, Release 1, Level 1.0
     Client date/time: 11/03/14   06:53:19
   (c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

   Node Name: TSMCL1
   Session established with server TSMGPFS1: AIX
     Server Version 7, Release 1, Level 0.0
     Server date/time: 11/03/14   06:53:20  Last access: 11/03/14   06:52:34

   Accessing as node: GPFS_PROXY

   Incremental backup of volume '/backup/'
   Normal File-->           3,135 /backup/citScanOutput.xml [Sent]
   ```

```
Normal File-->               0 /backup/dsmerror.log [Sent]
Expiring-->             10,060 /backup/gpfs1.bkp.cfg [Sent]
Normal File-->           1,596 /backup/mmdf.gpfs1 [Sent]
Normal File-->             424 /backup/mmlsnsd [Sent]
Normal File-->             918 /backup/mmlsnsd_M [Sent]
Successful incremental backup of '/backup/*'


Total number of objects inspected:           8
Total number of objects backed up:           5
Total number of objects updated:             0
Total number of objects rebound:             0
Total number of objects deleted:             0
Total number of objects expired:             1
Total number of objects failed:              0
Total number of objects encrypted:           0
Total number of objects grew:                0
Total number of retries:                     0
Total number of bytes inspected:         37.11 KB
Total number of bytes transferred:        6.05 KB
Data transfer time:                       0.00 sec
Network data transfer rate:         201,985.67 KB/sec
Aggregate data transfer rate:             2.06 KB/sec
Objects compressed by:                       0%
Total data reduction ratio:              83.68%
Elapsed processing time:                00:00:02
```

2. Back up the Spectrum Scale file system configuration information.

   In addition to the disks, the file system built on those volumes has configuration information that can be captured by using the **mmbackupconfig** command. This information includes: block size, replication factors, number and size of disks, storage pool layout, filesets and junction points, policy rules, quota information, and a number of other file system attributes. The file system configuration information can be backed up into a single file. In our scenario, we back up the configuration to a file and save the file to Tivoli Storage Manager by using the Tivoli Storage Manager BA client. See Example 7-80 for the operations that are performed to save the Spectrum Scale file system configuration data.

Example 7-80 Backing up the file system configuration for gpfs1

```
root@tsmcl1:/>mmbackupconfig gpfs1 -o /backup/gpfs1.bkp.cfg

mmbackupconfig: Processing file system gpfs1 ...
mmbackupconfig: Command successfully completed

root@tsmcl1:/>dsmc sel /backup/gpfs1.bkp.cfg
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   06:54:32
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

Node Name: TSMCL1
Session established with server TSMGPFS1: AIX
```

```
  Server Version 7, Release 1, Level 0.0
  Server date/time: 11/03/14   06:54:32  Last access: 11/03/14   06:53:21

Accessing as node: GPFS_PROXY
Selective Backup function invoked.

Normal File-->           10,083 /backup/gpfs1.bkp.cfg [Sent]
Selective Backup processing of '/backup/gpfs1.bkp.cfg' finished without failure.


Total number of objects inspected:           1
Total number of objects backed up:           1
Total number of objects updated:             0
Total number of objects rebound:             0
Total number of objects deleted:             0
Total number of objects expired:             0
Total number of objects failed:              0
Total number of objects encrypted:           0
Total number of objects grew:                0
Total number of retries:                     0
Total number of bytes inspected:          9.84 KB
Total number of bytes transferred:        9.87 KB
Data transfer time:                       0.00 sec
Network data transfer rate:       1,234,863.28 KB/sec
Aggregate data transfer rate:             8.15 KB/sec
Objects compressed by:                       0%
Total data reduction ratio:               0.00%
Elapsed processing time:              00:00:01
```

3. Premigrate all newer file data into auxiliary storage (Tivoli Storage Manager).

   File contents in a space-managed file system reside in the auxiliary storage, managed by HSM. In the case of Tivoli Storage Manager HSM, disk and tape pools on the Tivoli Storage Manager server typically hold the offline images of migrated files. HSM can also be used to premigrate all newer file data into auxiliary storage so that all files have either a migrated or premigrated status (XATTR) recorded, and their current contents are copied or updated into the auxiliary storage. The Tivoli Storage Manager **dsmmigrate** command can be used as in Example 7-81.

Example 7-81 Premigrate all files in file system to Tivoli Storage Manager

```
root@tsmcl1:/>dsmmigrate -Recursive -Premigrate /gpfs1
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   07:16:39
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

ANS9951I File: /gpfs1/tst.10 has already been migrated.
ANS9951I File: /gpfs1/tst.11 has already been migrated.
ANS9951I File: /gpfs1/tst.3 has already been migrated.
ANS9951I File: /gpfs1/tst.4 has already been migrated.
ANS9951I File: /gpfs1/tst.5 has already been migrated.
ANS9951I File: /gpfs1/tst.9 has already been migrated.
ANS1898I ***** Processed         3 files *****
ANS1898I ***** Processed         4 files *****
```

This operation copies the current resident files from gpfs1 file system to the auxiliary storage (HSM) so that the files can be recovered from Tivoli Storage Manager in case of recovery of the entire file system.

4. Create a snapshot of the current file system image by using **mmcrsnapshot**, for providing a quiescent backup image of the file system metadata. See Example 7-82.

Example 7-82 Creating a snapshot of the file system gpfs1 after premigration

```
root@tsmcl1:/>mmcrsnapshot gpfs1 gpfs1.snap
Flushing dirty data for snapshot gpfs1.snap...
Quiescing all file system operations.
Snapshot gpfs1.snap created with id 1.
```

5. Issue the **mmimgbackup** command to save an image of the file system metadata. By default, the image files are saved to the .imgbackup directory located by default in the root of the Spectrum Scale file system, unless a different location is specified using the '**-g <directory>**' flag. It can also be sent to Tivoli Storage Manager if configured. In our case, we save this image to Tivoli Storage Manager, as shown in Example 7-83. The metadata image is taken from the snapshot operation performed in the previous step.

Example 7-83 Backing up the file system metadata to Tivoli Storage Manager

```
root@tsmcl1:/>mmimgbackup gpfs1 -S gpfs1.snap -g /backup/ --tsm
mmimgbackup: [I] Image backup of /dev/gpfs1 begins at Mon Nov  3 07:27:14 EST 2014.
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name                 KB_Occupied        KB_Total  Percent_Occupied
system                       31098112       209715200     14.828735352%
[I] 4080 of 225408 inodes used: 1.810051%.
[I] Loaded policy rules from /backup//10748018/.imagePolicy.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-11-03@12:27:14 UTC
Parsed 2 policy rules.
     /* Automatically generated policy file for image backup */
RULE 'xl-d'  EXTERNAL LIST xl EXEC ''
RULE 'xl-m'  LIST xl directories_plus

[I] 2014-11-03@12:27:16.922 Directory entries scanned: 43.
[I] Directories scan: 35 files, 8 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-11-03@12:27:16.954 Sorting 43 file list records.
[I] Source: fs "gpfs1" snap "gpfs1.snap" path "/gpfs1/.snapshots/gpfs1.snap"
[I] Inodes scan: 35 files, 8 directories, 0 other objects, 0 'skipped' files and/or errors.
[I] 2014-11-03@12:27:17.015 Policy evaluation. 43 files scanned.
[I] Summary: 43 inodes scanned, 43 records written, 0 n/a, 0 errors.
mmimgbackup: [I] Image backup of /dev/gpfs1 ends at Mon Nov  3 07:27:17 EST 2014.
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   07:27:17
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

Node Name: TSMCL1
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 11/03/14   07:27:18  Last access: 11/03/14   07:16:40

Accessing as node: GPFS_PROXY
Selective Backup function invoked.
```

```
Directory-->              4,096 /backup/10748018 [Sent]
Normal File-->              136 /backup/10748018/.imagePolicy [Sent]
Directory-->                256 /backup/10748018/mmPolicy.9371898.DA6C46CD [Sent]
Normal File-->            1,088 /backup/10748018/mmPolicy.9371898.DA6C46CD/ImageArchive_20141103_12.27.14.idx
[Sent]
Normal File-->           20,160
/backup/10748018/mmPolicy.9371898.DA6C46CD/ImageArchive_20141103_12.27.14_000.sbr [Sent]
Selective Backup processing of '/backup/10748018/*' finished without failure.


Total number of objects inspected:           5
Total number of objects backed up:           5
Total number of objects updated:             0
Total number of objects rebound:             0
Total number of objects deleted:             0
Total number of objects expired:             0
Total number of objects failed:              0
Total number of objects encrypted:           0
Total number of objects grew:                0
Total number of retries:                     0
Total number of bytes inspected:         25.13 KB
Total number of bytes transferred:       20.97 KB
Data transfer time:                       0.00 sec
Network data transfer rate:         228,037.95 KB/sec
Aggregate data transfer rate:            10.19 KB/sec
Objects compressed by:                       0%
Total data reduction ratio:              16.53%
Elapsed processing time:                00:00:02
mmimgbackup: Successful backup up image files from directory /backup//10748018.
```

The metadata of the file system, the directories, inodes, attributes, symlinks, and so on are all captured in parallel by using the archive module extension feature of the **mmapplypolicy** command. After completing the parallel execution of the policy-driven archiving process, a collection of image files in this format will remain. These image files are gathered by the **mmimgbackup** command and archived to Tivoli Storage Manager automatically.

6. After the **mmimgbackup** is complete, delete the snapshot from the file system. See Example 7-84 for our scenario.

Example 7-84 Delete the Spectrum Scale snapshot after mmimgbackup

```
root@tsmcl1:/>mmdelsnapshot gpfs1 gpfs1.snap
Invalidating snapshot files in gpfs1.snap...
Deleting files in snapshot gpfs1.snap...
 100.00 % complete on Mon Nov  3 07:32:08 2014  (   225408 inodes with total
0 MB data processed)
Invalidating snapshot files in gpfs1.snap/F/...
Delete snapshot gpfs1.snap complete, err = 0
```

Note: Steps 1 and 2 should be performed regularly and on all cluster and file system configuration changes (new pools, disks added and removed, parameters changed, and so on). Steps 3 - 6 need to be repeated based on the required RPO (daily, hourly, weekly, and so on). If an unattended failure occurs on the GPFS cluster, recovery can be performed at the point in time of the latest premigrated operation performed.

Schedule 3 - 6 operations to be performed automatically based on the RPO requirements.

Recovery operations with SOBAR

This section provides a detailed example of the restore procedure with SOBAR, based on the previous backup scenario. In order to restore a file system, the configuration data stored from a previous run of **mmbackupconfig** and the image files produced from **mmimgbackup** must be accessible. In our scenario, all the recovery information including the Spectrum Scale cluster configuration (mmsdrfs), disk configuration, **mmbackupconfig** and **mmimgbackup** files are retrieved from Tivoli Storage Manager.

In our recovery scenario, we build a new Spectrum Scale cluster with two AIX nodes: fsaix1 and fsaix2, and restore the entire file system configuration for the previously backed up file system `gpfs1`. We assume that all infrastructure prerequisites (operating system, TCP/IP configuration, storage) are in place, including the SSH passwordless setup, and IBM Spectrum Scale 4.1.0.4 code is installed on the AIX nodes.

The following steps are performed for recovery:

1. Perform the Tivoli Storage Manager HSM installation and configuration. In our environment, we use migration to HSM based on Spectrum Scale policy, so before installing HSM for IBM Spectrum Scale, we disable the HSM automatic migration daemons (*dsmmonitord* and *dsmscoutd*) by exporting the following environment variable:

   ```
   export HSMINSTALLMODE=SCOUTFREE
   ```

   Install the Tivoli Storage Manager HSM software, including Tivoli Storage Manager BA client for IBM Spectrum Scale. For more details, see the following site:

   http://ibm.co/1OZt7Kk

   We use Tivoli Storage Manager HSM 7.1.1 on AIX 7.1, and the filesets on each node after the installation process are shown in Example 7-85.

Example 7-85 Tivoli Storage Manager HSM installed AIX filesets

```
root@fsaix1:/work>lslpp -L "tivoli.tsm*"
  Fileset                      Level  State  Type  Description (Uninstaller)
  ----------------------------------------------------------------------------
  tivoli.tsm.client.api.64bit
                             7.1.1.0    C     F    TSM Client - 64bit Application
                                                   Programming Interface
  tivoli.tsm.client.ba64.gpfs.base
                             7.1.1.0    C     F    TSM Client - Backup/Archive
                                                   Base Files
  tivoli.tsm.client.ba64.gpfs.common
                             7.1.1.0    C     F    TSM Client - Backup/Archive
                                                   Common Files
  tivoli.tsm.client.ba64.gpfs.image
                             7.1.1.0    C     F    TSM Client - IMAGE Backup
                                                   Client
  tivoli.tsm.client.ba64.gpfs.web
```

```
                            7.1.1.0    C     F    TSM Client - Backup/Archive
                                                  Java GUI & WEB Client
  tivoli.tsm.client.hsm.gpfs
                            7.1.1.0    C     F    TSM Client - Hierarchical
                                                  Storage Management


State codes:
 A -- Applied.
 B -- Broken.
 C -- Committed.
 E -- EFIX Locked.
 O -- Obsolete.  (partially migrated to newer version)
 ? -- Inconsistent State...Run lppchk -v.

Type codes:
 F -- Installp Fileset
 P -- Product
 C -- Component
 T -- Feature
 R -- RPM Package
 E -- Interim Fix
```

We configure the connection to the Tivoli Storage Manager server on each node (fsaix1 and fsaix2) by using the `dsm.opt` and `dsm.sys` files in the installation directory: `/usr/tivoli/tsm/client/ba/bin64`, as shown in Example 7-86.

Example 7-86 Configuration files for the Tivoli Storage Manager client

```
root@fsaix1:/usr/tivoli/tsm/client/ba/bin64>cat dsm.opt
SErvername tsmgpfs1
HSMGROUPEDMIGRATE YES
HSMEXTOBJIDATTR YES
HSMDISABLEAUTOMIGDAEMONS YES

root@fsaix1:/usr/tivoli/tsm/client/ba/bin64>cat dsm.sys

SErvername  tsmgpfs1
   COMMMethod         TCPip
   TCPPort            1500
   TCPServeraddress   172.16.20.153
   PASSwordaccess       generate
   NODENAME fsaix1                          ===> we use fsaix2 on node fsaix2
   ASNODENAME   GPFS_PROXY
```

We use the proxy nodename GPFS_PROXY as in the initial environment shown in Example 7-77 on page 392. This node owns the data saved by the old Spectrum Scale cluster, including the premigrated HSM files. On the Tivoli Storage Manager server, we register the two nodes in the same policy domain as node GPFS_PROXY and grant access for the two agent nodes, fsaix1 and fsaix2, to the proxy node. See Example 7-87 on page 400.

Example 7-87 Registering the new Spectrum Scale nodes and granting access to the proxy node

```
tsm: TSMGPFS1>q node GPFS_PROXY

Node Name                     Platform     Policy Domain      Days Since     Days Since     Locked?
                                           Name               Last Acce-       Password
                                                                     ss            Set
-------------------------     --------     --------------     ----------     ----------     -------
GPFS_PROXY                    AIX          GPFS_DOM                  <1             10       No

tsm: TSMGPFS1>register node fsaix1 XXXXX domain=GPFS_DOM maxnummp=4
ANR2060I Node FSAIX1 registered in policy domain GPFS_DOM.
ANR2099I Administrative userid FSAIX1 defined for OWNER access to node FSAIX1.

tsm: TSMGPFS1>register node fsaix2 XXXXX domain=GPFS_DOM maxnummp=4
ANR2060I Node FSAIX2 registered in policy domain GPFS_DOM.
ANR2099I Administrative userid FSAIX2 defined for OWNER access to node FSAIX2.

tsm: TSMGPFS1>grant proxy target=GPFS_PROXY agent=fsaix1,fsaix2
ANR0140I GRANT PROXYNODE: success.  Node FSAIX1 is granted proxy authority to node GPFS_PROXY.
ANR0140I GRANT PROXYNODE: success.  Node FSAIX2 is granted proxy authority to node GPFS_PROXY.

tsm: TSMGPFS1>q proxy

Target Node         Agent Node
---------------     ---------------------------------------------
GPFS_PROXY          TSMCL1 TSMCL2 FSAIX1 FSAIX2
```

Use **dsmc** on each Spectrum Scale node to log in to the Tivoli Storage Manager server for the first time, and save the password locally.

2. Restore all Spectrum Scale general configuration information, including **mmsdrfs**, nsd data, and mmbackup information from Tivoli Storage Manager. In our scenario, we saved this information in the /backup directory on one of the original cluster nodes. We restore this information in the same directory on one of the new cluster nodes. See Example 7-88.

Example 7-88 Restoring the Spectrum Scale cluster information

```
root@fsaix1:/>dsmc restore -sub=yes /backup/
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   07:42:36
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

Node Name: FSAIX1
Session established with server TSMGPFS1: AIX
  Server Version 7, Release 1, Level 0.0
  Server date/time: 11/03/14   07:42:36  Last access: 11/03/14   07:41:59

Accessing as node: GPFS_PROXY
Restore function invoked.

Restoring           4,096 /backup/10748018 [Done]
Restoring             256 /backup/10748018/mmPolicy.9371898.DA6C46CD [Done]
Restoring               0 /backup/dsmerror.log [Done]
Restoring           3,135 /backup/citScanOutput.xml [Done]
Restoring           1,596 /backup/mmdf.gpfs1 [Done]
Restoring             424 /backup/mmlsnsd [Done]
```

```
Restoring             918 /backup/mmlsnsd_M [Done]
Restoring          10,083 /backup/gpfs1.bkp.cfg [Done]
Restoring             136 /backup/10748018/.imagePolicy [Done]
Restoring           1,088
/backup/10748018/mmPolicy.9371898.DA6C46CD/ImageArchive_20141103_12.27.14.idx
[Done]
Restoring          20,160
/backup/10748018/mmPolicy.9371898.DA6C46CD/ImageArchive_20141103_12.27.14_000.sbr
[Done]
Restoring           3,807 /backup/mmsdrfs [Done]

Restore processing finished.

Total number of objects restored:            12
Total number of objects failed:               0
Total number of bytes transferred:        40.66 KB
Data transfer time:                        0.00 sec
Network data transfer rate:          142,195.69 KB/sec
Aggregate data transfer rate:             13.07 KB/sec
Elapsed processing time:                00:00:03
```

3. Inspect the `mmsdrfs` file for determining the original cluster configurations and settings. We build the new Spectrum Scale cluster with two nodes, fsaix1 and fsaix2, based on the original configuration of the Spectrum Scale cluster, saved in the `mmsdrfs` file, adapted to the new cluster environment. See the output of **`mmcrcluster`** in Example 7-89. In this example, we use the input file `nodes.hsm` with the following content:

   ```
   fsaix1:quorum-manager:
   fsaix2:quorum-manager:
   ```

Example 7-89 Creating the new Spectrum Scale cluster

```
root@fsaix1:/work> mmcrcluster -N /work/nodes.hsm --ccr-enable -r /usr/bin/ssh -R
/usr/bin/scp -C hsmgpfs -A
mmcrcluster: Performing preliminary node verification ...
mmcrcluster: Processing quorum and other critical nodes ...
mmcrcluster: Finalizing the cluster data structures ...
mmcrcluster: Command successfully completed
mmcrcluster: Warning: Not all nodes have proper GPFS license designations.
    Use the mmchlicense command to designate licenses as needed.
mmcrcluster: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

We apply the license designations for the nodes. Based on their roles, we use the server license for both nodes. See Example 7-90 for applying the license designations in our scenario.

Example 7-90 Applying the license designations to the cluster nodes

```
root@fsaix1:/work>mmchlicense server --accept -N all

The following nodes will be designated as possessing GPFS server licenses:
        fsaix2
        fsaix1
mmchlicense: Command successfully completed
mmchlicense: Propagating the cluster configuration data to all
```

```
  affected nodes.  This is an asynchronous process.
```

4. Determine the existing disk attached and their sizes and create the NSD for the tiebreaker disk. We inspect the content of the `/backup/mmdf.gpfs1` file (see the output in Example 7-74 on page 391) and evaluate the disks in the new environment and their sizes. In Example 7-91, we run a script for showing the new disks in their environment and the associated disk size.

Example 7-91 Disk and their sizes

```
root@fsaix1:/work>lspv
hdisk0          00f6f5d0d1fb5266                    rootvg          active
hdisk1          none                                None
hdisk2          none                                None
hdisk3          none                                None
hdisk4          none                                None
hdisk5          none                                None

root@fsaix1:/work>i=0; while [ $i -le 5 ]; do echo "hdisk$i : `bootinfo -s
hdisk$i`"; let i=i+1;done
hdisk0 : 35840                          => rootvg
hdisk1 : 1024                           => GPFS tiebreaker
hdisk2 : 10240                          => gpfs1 fs metadata
hdisk3 : 10240                          => gpfs1 fs metadata
hdisk4 : 102400                         => gpfs1 fs data
hdisk5 : 102400                         => gpfs1 fs data
```

The following input file is used for the **mmcrnsd** command for defining the nsd for the tiebreaker disk:

```
%nsd: device=hdisk1 usage=descOnly nsd=tie1nsd
```

We run the **mmcrnsd** command with the input file `tie.nsd` containing the preceding lines. See Example 7-92.

Example 7-92 Creating NSDs in the new Spectrum Scale cluster

```
root@fsaix1:/work>mmcrnsd -F ./tie.nsd
mmcrnsd: Processing disk hdisk1
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

5. Change the quorum configuration for using the tiebreaker nsd *tie1nsd*, defined in the previous step. See Example 7-93.

Example 7-93 Changing the cluster configuration for using the tiebreaker disk

```
root@fsaix1:/work>mmchconfig tiebreakerDisks=tie1nsd
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. Apply all specific configuration parameters from the original Spectrum Scale cluster. Inspect the `mmsdrfs` file restored from the original cluster. Always adjust the parameters according to the existing environment (as an example, pagepool cannot exceed half of the physical RAM of the node). Example 7-94 on page 403 shows the output of the **mmchconfig** command for applying the original settings. The applied settings are just examples of restoring of the original configuration and not a tuning recommendation.

Example 7-94 Customizing the cluster configuration with the parameters from the original cluster

```
root@fsaix1:/work>mmchconfig pagepool=1500M,prefetchThreads=100
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

7. Start the Spectrum Scale services by using: **mmstartup -a**.
8. Create the NSDs for the gpfs1 file system. We use **mmrestoreconfig** with the **-F <querycfg_file>** option to generate the NSD and file system recovery information in a file as shown in Example 7-95.

Example 7-95 Generating the NSD and file system recovery file

```
root@fsaix1:/backup>mmrestoreconfig gpfs1 -i gpfs1.bkp.cfg -F gpfs1.queryconfig
mmrestoreconfig: Configuration file successfully created in gpfs1.queryconfig
mmrestoreconfig: Command successfully completed

root@fsaix1:/backup>cat gpfs1.queryconfig
## ************************************************************
## Filesystem configuration file backup for file system: gpfs1
## Date Created: Sat Nov  1 19:33:35 EDT 2014
##
## The '#' character is the comment character.  Any parameter
## modified herein should have any preceding '#' removed.
## ************************************************************

######## NSD configuration #################
## Disk descriptor format for the mmcrnsd command.
## Please edit the disk and desired name fields to match
## your current hardware settings.
##
## The user then can uncomment the descriptor lines and
## use this file as input to the -F option.
#
# %nsd:
#   device=DiskName
#   usage=dataOnly
#   failureGroup=-1
#   pool=system
#
# %nsd:
#   device=DiskName
#   usage=dataOnly
#   failureGroup=-1
#   pool=system
.............................
```

The stanzas for **mmcrnsd** can be found in the queryconfig file as shown in the previous output example. We add the devices in the ‘device=’ lines based on the sizes of the disks and their roles identified in Step 4 (See Example 7-91 on page 402), so the resulted file nsd.hsm for **mmcrnsd** is shown in Example 7-96.

Example 7-96 Stanza file for re-creating the NSDs for the gpfs1 file system

```
root@fsaix1:/work>cat nsd.hsm
```

```
%nsd:
  device=hdisk4
  usage=dataOnly
  failureGroup=-1
  pool=system

%nsd:
  device=hdisk5
  usage=dataOnly
  failureGroup=-1
  pool=system

%nsd:
  device=hdisk2
  usage=metadataOnly
  failureGroup=-1
  pool=system

%nsd:
  device=hdisk3
  usage=metadataOnly
  failureGroup=-1
  pool=system

%pool:
  pool=system
  blockSize=262144
  usage=dataAndMetadata
  layoutMap=cluster
  allowWriteAffinity=no
```

We create the NSD devices by using the **mmcrnsd** command as shown in Example 7-97.

Example 7-97 Creating the NSDs for file system gpfs1

```
root@fsaix1:/work>mmcrnsd -F /work/nsd.hsm
mmcrnsd: Processing disk hdisk4
mmcrnsd: Processing disk hdisk5
mmcrnsd: Processing disk hdisk2
mmcrnsd: Processing disk hdisk3
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

9. Re-create the gpfs1 file system. We use the queryconfig file generated by **mmrestoreconfig** in Example 7-95 on page 403. The **mmcrfs** command is present under the “File System Configuration” section of the queryconfig file. We run the command with the indicated parameters to restore the file system with the original parameters at creation time. We change the name of the device to gpfs1 and replace the “NSD_DISK” field with the nsd.hsm file resulting from the previous step. See Example 7-98.

Example 7-98 Creating the file system gpfs1 with the original parameters

```
root@fsaix1:/backup>cat /backup/gpfs1.queryconfig | grep mmcrfs
# mmcrfs FS_NAME NSD_DISKS -i 4096 -j cluster --version 4.1.0.4 -z yes -L 4194304
-T /gpfs1 --inode-limit 225408:225408
```

```
root@fsaix1:/backup>mmcrfs gpfs1 -F /work/nsd.hsm -i 4096 -j cluster --version
4.1.0.4 -z yes -L 4194304 -T /gpfs1 --inode-limit 225408:225408

The following disks of gpfs1 will be formatted on node fsaix2:
    gpfs1nsd: size 102400 MB
    gpfs2nsd: size 102400 MB
    gpfs3nsd: size 10240 MB
    gpfs4nsd: size 10240 MB
Formatting file system ...
Disks up to size 1.1 TB can be added to storage pool system.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Observe in the previous output example, the presence of the **-z yes** option required for HSM enablement of the Spectrum Scale file system `gpfs1`.

10. Restore the file system attributes from the `mmbackupconfig` file. The **mmrestoreconfig** command is issued with the **--image-restore** option. See Example 7-99.

Example 7-99 Restoring the file system attributes for gpfs1

```
root@fsaix1:/backup>mmrestoreconfig gpfs1 -i /backup/gpfs1.bkp.cfg --image-restore
------------------------------------------------------
Configuration restore of gpfs1 begins at Mon Nov  3 07:55:19 EST 2014.
------------------------------------------------------
mmrestoreconfig: Checking disk settings for gpfs1:
mmrestoreconfig: Checking the number of storage pools defined for gpfs1.
mmrestoreconfig: Checking storage pool names defined for gpfs1.
mmrestoreconfig: Checking storage pool size for 'system'.

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:

mmrestoreconfig: Checking fileset configurations for gpfs1:
Fileset TestFS1 created with id 1 root inode 157696.
Fileset TestFS2 created with id 2 root inode 20608.

mmrestoreconfig: Checking policy rule configuration for gpfs1:
Restoring backed up policy file.
Validated policy `policyfile.backup': Parsed 3 policy rules.
Policy `policyfile.backup' installed and broadcast to all nodes.
mmrestoreconfig: Command successfully completed
```

Copy the hsm policy exec file from sample directory to `/var/mmfs/etc`, on both Spectrum Scale nodes:

```
cp -p /usr/lpp/mmfs/samples/ilm/mmpolicyExec-hsm.sample \
/var/mmfs/etc/mmpolicyExec-hsm.sample
chmod u+x /var/mmfs/etc/mmpolicyExec-hsm.sample
```

Add the callback entry for threshold migration on gpfs1. It can be found in the `mmsdrfs` file restored from the original cluster:

```
mmaddcallback MIGRATION --command /usr/lpp/mmfs/bin/mmstartpolicy --event
lowDiskSpace,noDiskSpace --parms "%eventName %fsName"
```

11. Mount the file system in read-only mode on the current cluster node for image restore:

```
mmmount gpfs1 -o ro
```

12. Perform the image restore based on the files restored in the `/backup` directory from Tivoli Storage Manager (see output of Example 7-88 on page 400). In our scenario, when we restored the entire `/backup` directory, the metadata image files were also restored. If there are multiple versions, select the latest version based on the format of `ImageArchive` files:

```
ImageArchive_YYYYMMDD_HH.mm.ss.idx,
ImageArchive_YYYYMMDD_HH.mm.ss_000.sbr
```

Example 7-100 shows the directory structure of the image files, restored from Tivoli Storage Manager.

Example 7-100 Selecting the latest version between multiple mmimgbackup file versions

```
root@fsaix1:/backup>ls -lR
total 72
drwxr-xr-x    3 root     system          256 Nov  3 07:27 10748018
-rw-rw-rw-    1 root     system         3135 Nov  3 06:53 citScanOutput.xml
-rw-r--r--    1 root     system            0 Nov  3 06:53 dsmerror.log
-rw-r--r--    1 root     system        10083 Nov  3 06:54 gpfs1.bkp.cfg
-rw-r--r--    1 root     system         2492 Nov  3 07:50 gpfs1.queryconfig
-rw-r--r--    1 root     system         1596 Nov  3 06:50 mmdf.gpfs1
-rw-r--r--    1 root     system          424 Nov  3 06:49 mmlsnsd
-rw-r--r--    1 root     system          918 Nov  3 06:50 mmlsnsd_M
-rw-r--r--    1 root     system         3208 Nov  3 07:58 mmsdrfs
./10748018:
total 8
-rw-r--r--    1 root     system          136 Nov  3 07:27 .imagePolicy
drwx------    2 root     system          256 Nov  3 07:27
mmPolicy.9371898.DA6C46CD

./10748018/mmPolicy.9371898.DA6C46CD:
total 48
-rw-r--r--    1 root     system         1088 Nov  3 07:27
ImageArchive_20141103_12.27.14.idx
-rw-r--r--    1 root     system        20160 Nov  3 07:27
ImageArchive_20141103_12.27.14_000.sbr
```

You need to provide the proper `mmPolicy` directory as parameter to the **mmimgrestore** command. In our case, we use **mmPolicy.9371898.DA6C46CD** (see Example 7-101).

Example 7-101 Restoring the metadata image file to gpfs1 file system

```
root@fsaix1:/backup>mmimgrestore gpfs1 /backup/10748018/mmPolicy.9371898.DA6C46CD
[I] GPFS Current Data Pool Utilization in KB and %
Pool_Name                    KB_Occupied        KB_Total  Percent_Occupied
system                            132096       209715200      0.062988281%
[I] 4038 of 225408 inodes used: 1.791418%.
[I] Loaded policy rules from /dev/null.
Evaluating policy rules with CURRENT_TIMESTAMP = 2014-11-03@13:02:23 UTC
Parsed 0 policy rules.
[W] Attention: It seems there are no effective nor useful rules. You may want to
terminate this command!
```

```
[I] Executing filelist: /dev/null...
[I] RESTORE:[I] Starting image restore pipeline
[I] RESTORE:[I] This task restored 43 inodes
[I] A total of 0 PDRs from filelist /dev/null have been processed; 0 'skipped'
records and/or errors.
[I] CONCLUDE:[I] Starting image restore pipeline
[I] A total of 0 files have been migrated, deleted or processed by an EXTERNAL
EXEC/script;
        0 'skipped' files and/or errors.
```

13. Unmount the file system *gpfs1*: **mmumount gpfs1**.

14. Restore quota configuration. If any quota enforcement was used in the prior file system, it can be restored now by using the **mmrestoreconfig** command. Example 7-102 shows how to run the command.

Example 7-102 Restore quota configuration

```
root@fsaix1:/backup>mmrestoreconfig gpfs1 -i gpfs1.bkp.cfg -Q only
------------------------------------------------------
Configuration restore of gpfs1 begins at Mon Nov  3 08:09:22 EST 2014.
------------------------------------------------------

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:
mmrestoreconfig: Command successfully completed
```

15. Mount the file system in read/write mode using the following command: **mmmount gpfs1 -a**.

16. Delete the HSM directory `.SpaceMan`. It contains file stubs from the former space management control information. This directory must be deleted before restarting HSM management. Use the following command: **rm -rf /gpfs1/.SpaceMan**.

17. Resume HSM management on the newly reconstructed file system `gpfs1`. We first restart the HSM services on both nodes by using the following command: **dsmmigfs restart**.

    To enable space management on the file system and permit recall of files, use the **dsmmigfs add** command as shown in Example 7-103.

Example 7-103 Adding space management to file system gpfs1

```
root@fsaix1:/>dsmmigfs add /gpfs1
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   08:24:58
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

Adding HSM support for /gpfs1 ...
ANS9087I Space management is successfully added to file system /gpfs1.
```

18. You can verify on the file status in the `gpfs1` file system by using the **dsmls** command, as shown in Example 7-104 on page 408. Observe in the output example that the files are in migrated state (*m*) because the files where premigrated as part of the backup procedure and the file system has been re-created.

Example 7-104 Listing the root directory of the /gpfs1 file system

```
root@fsaix1:/>dsmls /gpfs1
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   08:28:04
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

        ActS         ResS         ResB   FSt    FName

/gpfs1:
       <dir>         4096            0   -      .SpaceMan/
       <dir>         8192            1   -      .snapshots/
       <dir>         8192            8   -      TestFS1/
       <dir>         8192            8   -      TestFS2/
        3135            0            0   m      citScanOutput.xml
  7340032000            0            0   m      tst.10
  6291456000            0            0   m      tst.11
  6291456000            0            0   m      tst.12
  7340032000            0            0   m      tst.3
  7340032000            0            0   m      tst.4
  7340032000            0            0   m      tst.5
  7340032000            0            0   m      tst.9
```

For test purposes, we access the file `citScanOutput.xml` in the root directory of `/gpfs1` file system and then we check the file status by using the **`dsmls`** command. Observe in Example 7-105 that the file has changed to premigrated (p) status because during the read operation it was recalled from Tivoli Storage Manager.

Example 7-105 Accessing a file from file system /gpfs1

```
root@fsaix1:/gpfs1>cat citScanOutput.xml
<?xml version="1.0" encoding="UTF-8"?>
<Hardware>


        <Processor version="1" timestamp="1415013575">
                <Index IsKey="1">1</Index>
                <ID IsKey="1">1437f385ff059cf3796e72df7680938e</ID>
                <Family>Power PC Family</Family>
                <MaxClockSpeed>3300</MaxClockSpeed>
                <CurrentClockSpeed>3300</CurrentClockSpeed>
                <CPUEnabled>Y</CPUEnabled>
        </Processor>
.................................

root@fsaix1:/gpfs1>dsmls
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 7, Release 1, Level 1.0
  Client date/time: 11/03/14   08:54:51
(c) Copyright by IBM Corporation and other(s) 1990, 2014. All Rights Reserved.

        ActS         ResS         ResB   FSt    FName
```

```
/gpfs1:
       <dir>         4096            0   -      .SpaceMan/
       <dir>         8192            1   -      .snapshots/
       <dir>         8192            8   -      TestFS1/
       <dir>         8192            8   -      TestFS2/
        3135         3135            0   p      citScanOutput.xml
  7340032000            0            0   m      tst.10
  6291456000            0            0   m      tst.11
  6291456000            0            0   m      tst.12
  7340032000            0            0   m      tst.3
  7340032000            0            0   m      tst.4
  7340032000            0            0   m      tst.5
  7340032000            0            0   m      tst.9
```

8

Problem determination

This chapter describes an approach to problem determination that can guide you through the initial steps in problem determination and to help you to identify the component or subsystem that is experiencing a problem.

In addition to helping you determine the likely cause of the problem, this chapter describes some of the most common problems that can occur with IBM Spectrum Scale and, where possible, describes the actions you can take to resolve them.

For more problem determination commands and a full list of error messages and codes, refer to *GPFS V4.1: Problem Determination Guide*, GA76-0443-00.

Throughout this chapter, the term *Spectrum Scale* is used for clusters running GPFS version 3.5 or earlier. IBM Spectrum Scale applies only for version 4.1 or later.

This chapter contains the following topics:

- Problem determination process
- Gathering system information
- Operating system logs and IBM Spectrum Scale messages
- Verifying IBM Spectrum Scale cluster status
- Collecting IBM Spectrum Scale file system and disk information
- Collecting IBM Spectrum Scale general information and logs
- Collecting IBM Spectrum Scale debug information
- Working with IBM Spectrum Scale trace facility
- Deadlock detection features
- Additional information
- IBM Spectrum Scale problem determination scenarios:
  - Scenario 1: Spectrum Scale daemon not running
  - Scenario 2: Analyzing waiters
  - Scenario 3: Spectrum Scale file system hang
  - Scenario 4: Spectrum Scale file system unmounting
  - Scenario 5: Spectrum Scale file system not mounting
  - Scenario 6: Upgrading GPFS or Spectrum Scale
  - Scenario 7: NSD failures

© Copyright IBM Corp. 2015. All rights reserved.

8.1 Problem determination process

Maybe one of the most essential parts of the problem determination process is to really understand the nature of the problem. Some basic questions about how the problem was detected, how it is affecting the system, how often it occurs, can either determine that the problem is not GPFS or Spectrum Scale-related or can help during the further steps on the problem determination process. This section describes some of the questions that can help the administrators to better understand the problem and the paths to solve the most common issues that are related directly or indirectly to the Spectrum Scale. All the Spectrum Scale problem determination tasks performed in this book can be used to diagnose problems on clusters running GPFS 3.5 or earlier versions.

Spectrum Scale has its own error log, but the operating system error log is also useful because it contains information about hardware failures and operating system or other software failures that can affect Spectrum Scale.

Spectrum Scale also provides a system snapshot dump, trace, and other utilities that can be used to obtain detailed information about specific problems.

The problem determination tools provided with Spectrum Scale are intended for use by experienced system administrators who know how to collect data and run debugging routines.

The Spectrum Scale commands are located under the `/usr/lpp/mmfs/bin/` directory so it is suggested to add this path to the PATH environment variable.

Note: Spectrum Scale error logs and messages contain the MMFS prefix. This is intentional because Spectrum Scale shares many components with the IBM Multi-Media LAN Server, a related licensed program.

8.1.1 Understanding the problem

The first step in problem resolution is to clearly define the problem. It is important when trying to solve a problem to understand exactly what the users of the system perceive the problem to be. A clear definition of the problem is useful in some ways:

- It can give you a hint as to the cause of the problem.
- It allows you to demonstrate that the problem might be already resolved.
- It can lead to some other infrastructure components that are being used by the Spectrum Scale cluster (such as storage systems or network).

Usually the problem is first noticed by the user, and the system administrator can check system logs and environment overall state to ensure that the Spectrum Scale cluster is running and stable, but in order to better identify the root cause of the problem some questions might be asked to the user. The section 8.1.2, “Gathering information from the user” on page 412 explains some of the questions that might help the administrators to identify the problem.

8.1.2 Gathering information from the user

One of the best ways of understanding a problem from the user’s perspective is to ask the user some questions. This conversation can help you to deduce where the problem might be, and the time frame in which the user expects it to be resolved. This is important when analyzing large sets of data so the administrators can search for information during specific periods of time.

Sometimes the user's expectations might be beyond the scope of the system or the application that is running. Following are some usual questions when gathering information from the user during problem determination:

- What is the problem?
- How did the problem happen and how did it affect the users?
- What was the system doing? Or what was the system not doing?

After you determine the symptoms, try to establish the history of the problem by using some questions, such as:

- How did you first notice the problem? What operations were occurring?
- Did you do anything different that made you notice the problem? Any application changes?
- When did it happen?
- Try to identify and create a timeline so it can be used later when analyzing log files.
- Does it always happen at the same time, for example, when the same job or application runs?
- Does the same problem occur elsewhere?
- Is only one node experiencing the problem or are multiple nodes experiencing the same problem?
- Have any changes been made recently?
- This question refers to any type of change made to the system, ranging from adding new hardware or software, to configuration changes to existing software.
- If a change was made recently, were all of the prerequisites met before the change was made?

Software problems most often occur when changes are made to the system, and either the prerequisites have not been met, for example, system firmware is not at the minimum required level, or instructions were not followed properly.

Other changes, such as the addition of hardware can bring their own problems, such as cables incorrectly assembled, contacts bent, or addressing that is misconfigured.

The "How did you first notice the problem?" question might not help you directly, but it is useful in getting the person to talk about the problem. After starting to talk, the person invariably tells you things that can enable you to build a picture to help you to decide the starting point for problem resolution.

If the problem occurs on more than one node, look for similarities and differences between the situations.

8.2 Gathering system information

When looking at a Spectrum Scale cluster, consider that the scope of a system typically spans a cluster of computers that are connected through an interconnect (such as Ethernet) and have shared disks. Shared disk access is often through a storage area network (SAN) or a RAID array to provide direct disk access from multiple nodes. This approach means that collecting system data from any one of the nodes within the cluster (or clusters) might be necessary from the network subsystem and from the disk subsystem.

If the problem is not within Spectrum Scale, you likely need to refer to the platform-specific documentation to obtain additional system-data-collection procedures and to continue with system-problem determination.

8.2.1 Gathering operating system and IBM Spectrum Scale information

After gathering initial information from the user, the Spectrum Scale cluster administrator can use some basic operating system (OS) and Spectrum Scale commands to better document the problem and to check the overall status of the system, while providing pertinent information that might be used when calling the IBM service center, for example the version of the operating system and Spectrum Scale being used.

It is suggested to have the following information about your environment:

- Spectrum Scale command output with error.
- Application output (with timestamps, as possible).
- System config information (kernel version, NSD config, and so on).
- Copies of Spectrum Scale binary files.
- User data (copies of bad files, **`ls –li`** of problematic directories for example).

The information regarding the OS versions and Spectrum Scale levels also can be used to identify possible defects that are related to the Spectrum Scale product or even incompatibility issues between the OS and the Spectrum Scale.

- You can identify the OS version and fix level by running the appropriate command, **`oslevel -s`** for AIX environment, for example.
- Determine the Spectrum Scale level:
  - On AIX machines, run **`lslpp -l |grep gpfs`**. The output should be similar to Example 8-1.

Example 8-1 lslpp output from AIX machine

```
root@fsaix3:/>lslpp -l |grep gpfs
  gpfs.base                  4.1.0.0  COMMITTED  GPFS File Manager
  gpfs.crypto                4.1.0.0  COMMITTED  GPFS Cryptographic
  gpfs.ext                   4.1.0.0  COMMITTED  GPFS Extended Features
  gpfs.gskit                8.0.50.16  COMMITTED  GPFS GSKit Cryptography
  gpfs.base                  4.1.0.0  COMMITTED  GPFS File Manager
  gpfs.docs.data             4.1.0.0  COMMITTED  GPFS Server Manpages
```

  - On Linux RedHat, Ubuntu, or SUSE, run **`rpm -qa |grep gpfs`**. The output should be similar to Example 8-2.

Example 8-2 Using rpm to identify Spectrum Scale versions on SUSE or RedHat Linux

```
[root@fslinux1 4.1]# rpm -qa |grep gpfs
gpfs.base-4.1.0-0.ppc64
gpfs.gnr-4.1.0-0.ppc64
gpfs.msg.en_US-4.1.0-0.noarch
gpfs.docs-4.1.0-0.noarch
gpfs.ext-4.1.0-0.ppc64
gpfs.crypto-4.1.0-0.ppc64
gpfs.gpl-4.1.0-0.noarch
gpfs.gskit-8.0.50-16.ppc64
```

– On Debian Linux, run **dpkg -l | grep gpfs**. The output should be as shown in Example 8-3.

Example 8-3 Using dpkg to list the Spectrum Scale version

```
gpfs.base 4.1.0-0 GPFS File Manager
gpfs.docs 4.1.0-0 GPFS Server Manpages and Documentation
gpfs.gpl 4.1.0-0 GPFS Open Source Modules
gpfs.gskit 8.0.6-50 GPFS GSKit Cryptography Runtime
gpfs.msg.en_US 4.1.0-0 GPFS Server Messages - U.S. English
```

8.2.2 Testing connectivity and remote access

Some Spectrum Scale-related problems can be either related to the network or with authentication issues. Test the connectivity between the Spectrum Scale nodes:

- Ensure that all nodes are operating and responding to ping requests.

  Use commands such as **ping** or **netstat** to ensure that the network is up and available.

- Issue rsh/ssh on the nodes to make sure that all the nodes have remote access.

  If using the *adminMode=central* setup, make sure that all nodes in the Spectrum Scale cluster are able to communicate by using passwordless rsh and ssh. If *adminMode=central* is not used, you need to have the nodes, which the testing needs to be executed with passwordless access.

8.3 Operating system logs and IBM Spectrum Scale messages

Some operating system logs must be gathered during the problem determination process to identify infrastructure failures that potentially can affect the Spectrum Scale cluster, such as disk failures or network malfunctions. The Spectrum Scale generates information in its own logs but can also generate messages within the operating systems. This section describes some of the logs that can be collected to identify the possible problem or error that is affecting the Spectrum Scale environment.

8.3.1 Operating system logs

Each operating system has its own logs but the Spectrum Scale can also record file system or disk failures by using the error logging facility that is provided by the operating system:

- **syslog** facility on Linux
- **errpt** facility on AIX
- **Event Viewer** on Windows operating system

The error logging facility is referred to as the *error log* regardless of operating system-specific error log facility naming conventions.

- On AIX, the **errpt -a** command can display messages that are related to the Spectrum Scale cluster as shown in Example 8-4.

Example 8-4 errpt -a from an AIX Spectrum Scale node

```
LABEL:          MMFS_SYSTEM_UNMOUNT
IDENTIFIER:     C954F85D
```

```
Date/Time:       Fri Oct 10 16:36:09 2014
Sequence Number: 30
Machine Id:      00F6F5D04C00
Node Id:         fsaix1
Class:           S
Type:            PERM
WPAR:            Global
Resource Name:   mmfs

Description
STORAGE SUBSYSTEM FAILURE

Probable Causes
STORAGE SUBSYSTEM
COMMUNICATIONS SUBSYSTEM

Failure Causes
STORAGE SUBSYSTEM
COMMUNICATIONS SUBSYSTEM

        Recommended Actions
        CONTACT APPROPRIATE SERVICE REPRESENTATIVE

Detail Data
EVENT CODE
     3686577
STATUS CODE
         212
VOLUME
gpfs1
```

- On Linux nodes, the **`grep "mmfs:" /var/log/messages`** command can be executed to identify Spectrum Scale-related messages.
- On Windows operating system, use the *Event Viewer* and look for events with a source label of Spectrum Scale in the Application event category.

8.3.2 IBM Spectrum Scale messages

Since each operating system can receive some Spectrum Scale messages, previously mentioned as *Spectrum Scale error log*, several classes of events or errors can populate the OS logs. These classes are:

- “MMFS_ABNORMAL_SHUTDOWN”
- “MMFS_DISKFAIL”
- “MMFS_ENVIRON”
- “MMFS_FSSTRUCT”
- “MMFS_GENERIC”
- “MMFS_LONGDISKIO”
- “MMFS_QUOTA”
- “MMFS_SYSTEM_UNMOUNT”
- “MMFS_SYSTEM_WARNING”

MMFS_ABNORMAL_SHUTDOWN

The MMFS_ABNORMAL_SHUTDOWN error log entry means that Spectrum Scale has determined that it must shut down all Spectrum Scale operations on this node because of a problem. Insufficient memory on the node to handle critical recovery situations can cause this error. In general, there are other error log entries from Spectrum Scale or some other component that is associated with this error log entry.

MMFS_DISKFAIL

The MMFS_DISKFAIL error log entry indicates that Spectrum Scale has detected the failure of a disk and forced the disk to the stopped state. This is ordinarily not a Spectrum Scale error but a failure in the disk subsystem or the path to the disk subsystem.

MMFS_ENVIRON

MMFS_ENVIRON error log entry records are associated with other records of the MMFS_GENERIC or MMFS_SYSTEM_UNMOUNT types. They indicate that the root cause of the error is external to Spectrum Scale and usually in the network that supports Spectrum Scale. Check the network and its physical connections. The data portion of this record supplies the return code that is provided by the communications code.

MMFS_FSSTRUCT

The MMFS_FSSTRUCT error log entry indicates that Spectrum Scale has detected a problem with the on-disk structure of the file system. The severity of these errors depends on the exact nature of the inconsistent data structure. If it is limited to a single file, EIO errors are reported to the application and operation continues. If the inconsistency affects vital metadata structures, operation ceases on this file system. These errors are often associated with an MMFS_SYSTEM_UNMOUNT error log entry and will probably occur on all nodes. If the error occurs on all nodes, some critical piece of the file system is inconsistent. This can occur as a result of a Spectrum Scale error or an error in the disk system. If the file system is still mounted, unmount the file system everywhere, then collect data from the **mmfsck -n** command before calling IBM service.

MMFS_GENERIC

The MMFS_GENERIC error log entry means that Spectrum Scale self-diagnostics have detected an internal error or that additional information is being provided with an MMFS_SYSTEM_UNMOUNT report. If the record is associated with an MMFS_SYSTEM_UNMOUNT report, the event code fields in the records are the same. The error code and return code fields might describe the error.

If the error is generated by the self-diagnostic routines, service personnel should interpret the return and error code fields since the use of these fields varies by the specific error. Errors caused by the self-checking logic result in the shutdown of Spectrum Scale on this node.

MMFS_GENERIC errors can result from an inability to reach a critical disk resource. These errors might look different depending on the specific disk resource that has become unavailable, like logs and allocation maps. This type of error is usually associated with other error indications. Other errors that are generated by disk subsystems, high availability components, and communications components at the same time as, or immediately preceding the Spectrum Scale error, should be pursued first because they might be the cause of these errors. MMFS_GENERIC error indications without an associated error of those types represent a Spectrum Scale problem that requires the IBM Support Center.

MMFS_LONGDISKIO

The MMFS_LONGDISKIO error log entry indicates that Spectrum Scale is experiencing very long response time for disk requests. This is a warning message and can indicate that your disk system is overloaded or that a failing disk is requiring many I/O retries. Follow your operating system's instructions for monitoring the performance of your I/O subsystem on this node and on any disk server nodes that might be involved. The data portion of this error record specifies the disk involved. There might be related error log entries from the disk subsystems that will pinpoint the actual cause of the problem. The **mmpmon** command can be used to analyze I/O performance on a per-node basis.

MMFS_QUOTA

The MMFS_QUOTA error log entry is used when Spectrum Scale detects a problem in the handling of quota information. This entry is created when the quota manager has a problem reading or writing the quota file. If the quota manager cannot read all entries in the quota file when mounting a file system with quotas enabled, the quota manager shuts down but file system manager initialization continues. Mounts do not succeed and will return an appropriate error message. Quota accounting depends on a consistent mapping between user names and their numeric identifiers. This means that a single user accessing a quota-enabled file system from different nodes should map to the same numeric user identifier from each node. Within a local cluster, this is usually achieved by ensuring that /etc/passwd and /etc/group are identical across the cluster. When accessing quota-enabled file systems from other clusters, you need to either ensure that individual-accessing users have equivalent entries in /etc/passwd and /etc/group, or use the user identity mapping facility.

It might be necessary to run an offline quota check (**mmcheckquota**) to repair or re-create the quota file. If the quota file is corrupted, **mmcheckquota** will not restore it. The file must be restored from the backup copy. If there is no backup copy, an empty file can be set as the new quota file. This is equivalent to re-creating the quota file. To set an empty file or use the backup file, issue the **mmcheckquota** command with the appropriate option:

- *-u UserQuotaFilename* for the user quota file
- *-g GroupQuotaFilename* for the group quota file
- *-j FilesetQuotaFilename* for the fileset quota file

After replacing the appropriate quota file, reissue the **mmcheckquota** command to check the file system inode and space usage. For more information, see the IBM white paper entitled *UID Mapping for Spectrum Scale in a Multi-cluster Environment* in the IBM Knowledge Center at the following website:

http://ibm.co/1Ox2ku4

MMFS_SYSTEM_UNMOUNT

The MMFS_SYSTEM_UNMOUNT error log entry means that Spectrum Scale discovered a condition that might result in data corruption if operation with this file system continues from this node. Spectrum Scale marked the file system as disconnected and applications accessing files within the file system receive ESTALE errors. This can be the result of:

- The loss of a path to all disks containing a critical data structure.
- An internal processing error within the file system.

When this message is displayed, verify the storage subsystem for more errors.

MMFS_SYSTEM_WARNING

The MMFS_SYSTEM_WARNING error log entry means that Spectrum Scale has detected a system-level value approaching its maximum limit. This might occur as a result of the number

of inodes (files) reaching its limit. If so, issue the **mmchfs** command to increase the number of inodes for the file system so there is at least a minimum of 5% free.

8.4 Verifying IBM Spectrum Scale cluster status

After checking basic infrastructure components, verifying initial OS, and Spectrum Scale logs, the administrator can run some Spectrum Scale commands to identify the overall status of the Spectrum Scale cluster that might be helpful to solve the problem, or to better clarify the type of additional data that should be collected. In this chapter, we cover some commands that can be used during an initial problem determination analysis.

8.4.1 The mmgetstate command

The **mmgetstate** command shows the state of the Spectrum Scale daemon on one or more nodes. If any node state is down, the **mmstartup -N** command can be issued to the specific node that is down. The **mmgetstate** command can be used with the following options:

-a List all nodes in the GPFS cluster. The option does not display information for nodes that cannot be reached. You may obtain more information if you specify the **-v** option.

-L Additionally display quorum, number of nodes up, and total number of nodes. The total number of nodes can sometimes be larger than the actual number of nodes in the cluster. This is the case when nodes from other clusters have established connections for the purposes of mounting a file system that belongs to your cluster.

-s Display summary information: number of local and remote nodes that have joined in the cluster, number of quorum nodes, and so forth.

-v Display intermediate error messages.

When issuing the **mmgetstate** command, some node status can be displayed as shown in Example 8-5. Following are the states that are recognized and displayed by this command:

active GPFS is ready for operations.

arbitrating A node is trying to form quorum with the other available nodes.

down GPFS daemon is not running on the node or is recovering from an internal error.

unknown Unknown value. Node cannot be reached or some other error occurred.

Example 8-5 Issuing the mmgetstate command

```
root@fsaix1:/>/usr/lpp/mmfs/bin/mmgetstate -aL
Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
------------------------------------------------------------------------------------
       1      fsaix1             1        4          4       active      quorum node
       2      fsaix2             1        4          4       active      quorum node
       3      fsaix3             1        4          4       active      quorum node
       4      fsaix4             1        4          4       active      quorum node
```

8.4.2 The mmlsconfig command

The **mmlsconfig** command shows basic configuration information of the Spectrum Scale cluster.

When the **mmlsconfig** command has no specific attributes to be requested, the command displays all values that have been explicitly set by the user. Depending on your configuration, additional information that is set by Spectrum Scale may be displayed. Refer to Example 8-6.

Example 8-6 The mmlsconfig command output

```
root@fsaix1:/>/usr/lpp/mmfs/bin/mmlsconfig
Configuration data for cluster tsmgpfs.fsaix1:
----------------------------------------------
clusterName tsmgpfs.fsaix1
clusterId 12742445374757012402
dmapiFileHandleSize 32
minReleaseLevel 4.1.0.4
usePersistentReserve yes
ccrEnabled yes
autoload yes
tiebreakerDisks tie1
adminMode central
File systems in cluster tsmgpfs.fsaix1:
---------------------------------------
/dev/gpfs2
/dev/gpfs3
/dev/gpfs4
/dev/wpargpfs01
```

8.4.3 The mmlscluster command

The **mmlscluster** displays the current configuration information for a Spectrum Scale cluster as shown in Example 8-7. The command can be also used with the **--cnfs** flag to display information about *Clustered NFS.*

Example 8-7 The mmlscluster output

```
GPFS cluster information
========================
  GPFS cluster name:         tsmgpfs.fsaix1
  GPFS cluster id:           12742445374757012402
  GPFS UID domain:           tsmgpfs.fsaix1
  Remote shell command:      /usr/bin/rsh
  Remote file copy command:  /usr/bin/rcp
  Repository type:           CCR

 Node  Daemon node name  IP address     Admin node name  Designation
-----------------------------------------------------------------------
   1   fsaix1            172.16.20.153 fsaix1           quorum-manager
   2   fsaix2            172.16.20.154 fsaix2           quorum-manager
   3   fsaix3            172.16.20.155 fsaix3
```

8.5 Collecting IBM Spectrum Scale file system and disk information

The Spectrum Scale provides some useful commands to display information about the file systems that are currently being used by the cluster and NSD information. The information that is gathered by this command can be used for further problem determination procedures and help the administrators to have a more detailed view about the environment when contacting the IBM Service Center.

8.5.1 The mmlsfs command

The **mmlsfs** command lists the Spectrum Scale file systems that currently are part of the Spectrum Scale cluster.

Example 8-8 shows the **mmlsfs** command output.

Example 8-8 mmlsfs command output

```
root@fsaix1:/usr/lpp/mmfs/bin>mmlsfs GPFS
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 8192                     Minimum fragment size in bytes
 -i                 4096                     Inode size in bytes
 -I                 16384                    Indirect block size in bytes
 -m                 1                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 1                        Default number of data replicas
 -R                 2                        Maximum number of data replicas
 -j                 cluster                  Block allocation type
 -D                 nfs4                     File locking semantics in effect
 -k                 all                      ACL semantics in effect
 -n                 32                       Estimated number of nodes that will
mount file system
 -B                 262144                   Block size
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota No                       Per-fileset quota enforcement
 --filesetdf        No                       Fileset df enabled?
 -V                 14.10 (4.1.0.4)          File system version
 --create-time      Wed Oct 22 16:20:08 2014 File system creation time
 -z                 No                       Is DMAPI enabled?
 -L                 4194304                  Logfile size
 -E                 Yes                      Exact mtime mount option
 -S                 No                       Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           Yes                      Fast external attributes enabled?
 --encryption       No                       Encryption enabled?
 --inode-limit      65856                    Maximum number of inodes
 --log-replicas     0                        Number of log replicas
 --is4KAligned      Yes                      is4KAligned?
 --rapid-repair     Yes                      rapidRepair enabled?
 --write-cache-threshold 0                   HAWC Threshold (max 65536)
 -P                 system                   Disk storage pools in file system
```

```
 -d                 NSD001;NSD002;NSD003     Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /shared                  Default mount point
 --mount-priority   0                        Mount priority
```

8.5.2 The mmlsmount command

The **mmlsmount** command lists the Spectrum Scale nodes that have Spectrum Scale file systems mounted.

Use the **-L** flag to see the node name and IP address of each node that has the file system in use. This command can be used for all file systems, all remotely mounted file systems, or file systems that are mounted on nodes of certain clusters.

The **mmlsmount** command can be useful in some situations:

- To determine which nodes have a file system mounted
- When mounting a file system on multiple nodes to determine which nodes have successfully completed the mount and which nodes have failed
- To determine the extension of the problem when a file system is mounted, but appears to be inaccessible to some nodes but accessible to others
- To determine the affected nodes when a normal unmount has not completed
- To determine the affected nodes when a file system has force unmounted on some nodes but not others

Example 8-9 shows the **mmlsmount** output with the **-L** option for all nodes.

Example 8-9 The mmlsmount command

```
root@fsaix1:/.ssh>/usr/lpp/mmfs/bin/mmlsmount all -L

File system gpfs2 is mounted on 3 nodes:
  172.16.20.153   fsaix1                   (internal mount)
  172.16.20.155   fsaix3
  172.16.20.154   fsaix2

File system gpfs3 is mounted on 3 nodes:
  172.16.20.153   fsaix1                   (internal mount)
  172.16.20.155   fsaix3
  172.16.20.154   fsaix2

File system gpfs4 is mounted on 3 nodes:
  172.16.20.153   fsaix1                   (internal mount)
  172.16.20.155   fsaix3
  172.16.20.154   fsaix2

File system wpargpfs01 is mounted on 3 nodes:
  172.16.20.153   fsaix1                   (internal mount)
  172.16.20.155   fsaix3
  172.16.20.154   fsaix2
```

8.5.3 The mmlsnsd command

The **mmlsnsd** command displays information about the disks (NSDs) defined in the cluster. The command can be executed without parameters and the information as shown in Example 8-10 is displayed.

Example 8-10 The mmlsnsd command

```
root@fsaix1:/.ssh>/usr/lpp/mmfs/bin/mmlsnsd
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 gpfs2         gpfs7nsd     (directly attached)
 gpfs3         gpfs8nsd     (directly attached)
 gpfs4         gpfs9nsd     (directly attached)
 wpargpfs01    gpfs2nsd     (directly attached)
 (free disk)   aix_lv1      (directly attached)
 (free disk)   gpfs5nsd     (directly attached)
 (free disk)   gpfs6nsd     (directly attached)
 (free disk)   tie1         (directly attached)
```

It is possible to identify the association between the device names and the NSDs for all nodes by executing the **mmlsnsd** command with the **-M** option (uppercase m) as shown in Example 8-11.

Example 8-11 The mmlsnsd -M

```
root@fsaix1:/.ssh>/usr/lpp/mmfs/bin/mmlsnsd -M
Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------
 aix_lv1      AC101499543D385A   /dev/gpfs_lv   fsaix1
 aix_lv1      AC101499543D385A   -              fsaix2                   directly
attached
 aix_lv1      AC101499543D385A   -              fsaix3                   directly
attached
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix1
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix2
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix3
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix1
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix2
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix3
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix1
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix2
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix3
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix1
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix2
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix3
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix1
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix2
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix3
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix1
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix2
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix3
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix1
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix2
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix3
```

8.5.4 The mmwindisk command

When running Spectrum Scale on Windows operating system nodes, the **mmwindisk** command can be used to display information about the disks that are known to the operating system along with partitioning information relevant to Spectrum Scale.

If you issue the **mmwindisk** list, your output is similar to Example 8-12.

Example 8-12 Output from the mmwindisk command

```
Disk Avail Type     Status    Size        GPFS Partition ID
---- ----- ------- --------- --------    ------------------------------------
0           BASIC   ONLINE      137 GiB
1           GPFS    ONLINE      55 GiB    362DD84E-3D2E-4A59-B96B-BDE64E31ACCF
2           GPFS    ONLINE      200 GiB   BD5E64E4-32C8-44CE-8687-B14982848AD2
3           GPFS    ONLINE      55 GiB    B3EC846C-9C41-4EFD-940D-1AFA6E2D08FB
4           GPFS    ONLINE      55 GiB    6023455C-353D-40D1-BCEB-FF8E73BF6C0F
5           GPFS    ONLINE      55 GiB    2886391A-BB2D-4BDF-BE59-F33860441262
6     YES   UNALLOC  OFFLINE     55 GiB
7     YES   UNALLOC  OFFLINE     200 GiB
```

The following information is shown in the **mmwindisk** columns' output:

Disk Indicates the Windows operating system disk number as shown in the Disk Management console and the *DISKPART* command-line utility.

Avail Shows the value YES when the disk is available and in a state suitable for creating an NSD.

GPFS Partition ID Is the unique ID for the GPFS partition on the disk.

The **mmwindisk** command does not provide the NSD volume ID. The **mmlsnsd -m** is indicated to find the relationship between NSDs and devices, which are disk numbers on Windows operating systems.

8.5.5 The mmpmon command

When gathering disk-related data, the **mmpmon** command can be used to gather performance statistics from Spectrum Scale disks on the node that the command is executed. It is not primarily a diagnostic tool, but can be used as one for certain problems. For example, running **mmpmon** on several nodes can be used to detect nodes that are experiencing performance degradation or connectivity problems.

The **mmpmon** requires some options and an input file to run. Following are the available options:

-d IntegerDelayValue Specifies a number of milliseconds to sleep after one invocation of all the requests in the input file. The default value is 1000. This value must be an integer greater than or equal to 500, and less than or equal to 8,000,000.

-p Indicates to generate output that can be parsed by a script or program. If this option is not specified, human-readable output is produced.

-r IntegerRepeatValue Specifies the number of times to run all the requests in the input file. The default value is one. Specify an integer between zero and 8,000,000. Zero means to run forever, in which case processing

continues until it is interrupted. This feature is used, for example, by a driving script or application program that repeatedly reads the result from a pipe.

-s Indicates to suppress the prompt on input. Use of the **-i** flag implies use of the **-s** flag. For use in a pipe or with redirected input (<), the **-s** flag is preferred. If not suppressed, the prompts go to standard error (stderr).

-t IntegerTimeoutValue Specifies a number of seconds to wait for responses from the GPFS daemon before considering the connection to have failed. The default value is 60. This value must be an integer greater than or equal to 1, and less than or equal to 8,000,000.

After selecting the appropriate option, an input file is needed in order to determine the type of information that should be collected. The input file can have the following options:

fs_io_s Displays I/O statistics per mounted file system.

io_s Displays I/O statistics for the entire node.

nlist add name [name...]
Adds node names to a list of nodes for mmpmon processing.

nlist del Deletes a node list.

nlist new name [name...]
Creates a new node list.

nlist s Shows the contents of the current node list.

nlist sub name [name...]
Deletes node names from a list of nodes for mmpmon processing.

once request Indicates that the request is to be performed only once.

reset Resets statistics to zero.

rhist nr Changes the request histogram facility request size and latency ranges.

rhist off Disables the request histogram facility. This is the default.

rhist on Enables the request histogram facility.

rhist p Displays the request histogram facility pattern.

rhist reset Resets the request histogram facility data to zero.

rhist s Displays the request histogram facility statistics values.

ver Displays mmpmon version.

vio_s [f rg RecoveryGroupName [da DeclusteredArray [v Vdisk]]] [reset]
Displays GPFS Native RAID VDisk I/O statistics.

vio_s_reset [f rg RecoveryGroupName [da DeclusteredArray [v Vdisk]]]
Resets GPFS Native RAID VDisk I/O statistics.

To demonstrate the **mmpmon** usage, the information as shown in Example 8-13 was added to our test environment.

Example 8-13 Contents of the input file for the mmpmon command

```
root@fsaix2:/tmp/gpfs.snapOut/15204378>cat inputfile
ver
io_s
fs_io_s
```

After creating the input file, the **mmpmon** command was executed as shown in example Example 8-14.

Example 8-14 The mmpmon command

```
root@fsaix2:/tmp/gpfs.snapOut/15204378>mmpmon -i inputfile -r 1 -d 5000
mmpmon node 172.16.20.154 name fsaix2 version 3.5.4
mmpmon node 172.16.20.154 name fsaix2 io_s OK
timestamp:      1413412078/498079
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0
mmpmon node 172.16.20.154 name fsaix2 fs_io_s OK
cluster:        tsmgpfs.fsaix1
filesystem:     gpfs2
disks:                  1
timestamp:      1413412078/498222
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
```

```
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 172.16.20.154 name fsaix2 fs_io_s OK
cluster:        tsmgpfs.fsaix1
filesystem:     gpfs4
disks:                   1
timestamp:      1413412078/498222
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 172.16.20.154 name fsaix2 fs_io_s OK
cluster:        tsmgpfs.fsaix1
filesystem:     wpargpfs01
disks:                   1
timestamp:      1413412078/498222
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0
mmpmon node 172.16.20.154 name fsaix2 rhist off OK
```

Note: The **mmpmon** command is fully documented under the *Monitoring GPFS I/O performance with the* **mmpmon** *command topic* in the *GPFS V4.1: Advanced Administration Guide,* SC23-7032-00.

8.6 Collecting IBM Spectrum Scale general information and logs

In this section, we describe the types of Spectrum Scale logs that can be collected to start the problem determination process and some commands that can be executed to generate useful data for analysis.

8.6.1 Considerations when gathering Spectrum Scale information

Consider the following items when gathering data:

- From which nodes should I collect data? Which nodes are experiencing the problem?

  If this answer is not known, the best approach is to collect data from all nodes. In a large cluster, a sufficient way might be to start by collecting data from the manager nodes and the node experiencing the problem (when identified).

- Consider using the **mmdsh** command

  When collecting data from multiple nodes, use the **mmdsh command**. This command uses the administration communication method that is defined in the cluster to send any command out to the nodes.

  The **mmdsh** command requires passwordless remote-shell communications between the nodes on which you want to collect data. The **mmdsh** command requires the *working collective file* (WCOLL) that is a file that consists of node names, one per line, as illustrated in our lab environment, as shown in Example 8-15.

Example 8-15 WCOLL file contents sample

```
root@fsaix1:/gpfs>cat /etc/gpfs.nodes
fsaix1
fsaix2
fsaix3
fsaix4
```

After creating the WCOLL file, set the environment variable to reference the file containing the node names as shown in Example 8-16.

Example 8-16 Exporting the WCOLL environment variable

```
root@fsaix1:/gpfs>export WCOLL=/etc/gpfs.nodes
root@fsaix1:/gpfs>echo $WCOLL
/etc/gpfs.nodes
```

Once the variable is exported, the syntax of the **mmdsh** command to be used is as shown in Example 8-17.

Example 8-17 Sample of mmdsh command with WCOLL variable

```
root@fsaix1:/gpfs>/usr/lpp/mmfs/bin/mmdsh date
fsaix4:  Fri Oct 10 17:44:30 EDT 2014
fsaix2:  Fri Oct 10 17:44:30 EDT 2014
fsaix3:  Fri Oct 10 17:44:30 EDT 2014
fsaix1:  Fri Oct 10 17:44:30 EDT 2014
```

Without the WCOLL variable, the **mmdsh** command syntax is shown in Example 8-18.

Example 8-18 The mmdsh command to all nodes

```
root@fsaix1:/gpfs>/usr/lpp/mmfs/bin/mmdsh -N all date
tsmcl1:  Fri Oct 24 12:25:35 EDT 2014
tsmcl2:  Fri Oct 24 12:25:36 EDT 2014
tsmsrv:  Fri Oct 24 12:25:36 EDT 2014
fsaix4:  Fri Oct 24 12:25:36 EDT 2014
```

The **mmdsh -N all** command runs in all GPFS nodes and if nodes in your cluster are known to be down because some problem that occurred. The use of the collective file, including only the active nodes, can help you eliminate the timeout that **mmdsh** command experiences and reduce the time needed to collect the data needed. It is also possible to use **mmdsh -N node1, node2** instead of the WCOLL variable.

Note: The **mmdsh** command is not a full documented command and its use in this book is provided for data gathering purposes.

8.6.2 IBM Spectrum Scale severity tags

Spectrum Scale now assigns a severity level to most of its log messages. The severity level is visible as part of the message prefix and consists of a message severity code between square brackets. This additional information generated by the mmfsd daemon can help you to understand the criticality of the event. This type of additional information can also be referred to as *severity tags.*

The following severities can be assigned:

- **I - “informational”**

  Normal operation. This message by itself indicates no wrong.

- **W - “warning”**

  There is some problem, but command execution continues. The problem may be a transient inconsistency. It might be that the command skipped some operations on some objects, or is reporting an irregularity that may be of interest. For example, a multi-pass command operating on many files discovers during its second pass that a file that was present during the first pass is no longer present. Perhaps it was removed by another command or program.

- **E - “error”**

  An error. Command execution might or might not continue. Usually this error was likely caused by a persistent condition and remains until corrected by some other program or administrative action. For example, a command operating on a single file or other Spectrum Scale object may terminate upon encountering any condition of severity “E”. But as another example, a command operating on a list of files, finding that one of the files has permission bits set that disallow the operation, may continue to operate on all other files within said list of files.

- **S - “severe error”**

  The error is likely related to the state of the file system. Command execution is usually halted due to this kind of error. For example, the file system is not mounted, so the command cannot execute.

- **X - “exception”**

  The system has discovered an internal inconsistency of some kind. Command execution may be halted or the system might attempt to continue regardless of the inconsistency. These kinds of errors should be reported to IBM.

Optionally, the severity tag can include a numeric code:

- **nnn**

  When present, this is an error code associated with this message. If this was the only problem encountered by the command, the command return code would be nnn.

If a message has no severity tag, the message does not conform to this specification. The severity must be determined by examining the text and any supplemental information provided in the message catalog or by IBM technical support.

Example 8-19 shows the *severity tag* **I** being displayed, since they are informational messages only.

Example 8-19 Sample /var/adm/ras/mmfs.log.latest

```
Mon Nov 09:05:08.007 2014: [I] mmfsd initializing. {Version: 4.1.0.4 Built:
Oct  8 2014 10:46:32} ...
Mon Nov  3 09:05:08.022 2014: [I] Cleaning old shared memory ...
Mon Nov  3 09:05:08.050 2014: [I] First pass parsing mmfs.cfg ...
Mon Nov  3 09:05:08.058 2014: [I] Enabled automated deadlock detection.
Mon Nov  3 09:05:08.074 2014: [I] Enabled automated deadlock debug data collection.
Mon Nov  3 09:05:08.090 2014: [I] Initializing the main process ...
Mon Nov  3 09:05:08.104 2014: [I] Second pass parsing mmfs.cfg ...
Mon Nov  3 09:05:08.118 2014: [I] Initializing the page pool ...
Mon Nov  3 09:05:08.406 2014: [I] Initializing the mailbox message system ...
Mon Nov  3 09:05:08.433 2014: [I] Initializing encryption ...
Mon Nov  3 09:05:08.445 2014: [I] Encryption: loaded crypto library: IBM CryptoL
```

8.6.3 Spectrum Scale logs

Some Spectrum Scale messages can be generated within the OS error log facilities but the Spectrum Scale has its own logging mechanism to generate information about conditions or errors that have been detected on each node, as well as operational events such as file system mounts. Usually, the Spectrum Scale log is the first place to look when attempting to debug abnormal events. Since Spectrum Scale is a cluster file system, events that occur on one node might affect system behavior on other nodes, and all Spectrum Scale logs can have relevant data. The Spectrum Scale logs are formatted as follows:

- `/var/adm/ras` is the location directory
- `/var/adm/ras/mmfs.log.*` is the log name
  - All log file names incorporate host name and timestamp
  - `mmfs.log.latest` and `mmfs.log.previous` are symlinks
  - `/var/adm/ras/mmfs.log.latest` always contains the most updated information
- Logs are auto-rotated when Spectrum Scale is restarted by sysadmin

Gathering live data implies that you are currently experiencing a Spectrum Scale problem or that problem determination can occur only with data that is captured while the problem is occurring.

Gathering data after the failure and historical data (such as `mmfs.logs`, system error reports, and so on) can be gathered anytime after the problem occurred.

At Spectrum Scale start, log files that have not been accessed during the last 10 days are deleted. If you want to save old log files, copy them elsewhere.

Example 8-20 on page 431 shows normal operational messages that appear in the Spectrum Scale log file named `mmfs.log.2014.10.10.19.47.41.fsaix1`. The file format is "`mmsgs.log.timestamp.hostname`".

Example 8-20 Spectrum Scale mmfs.log.2014.10.10.19.47.41.fsaix1

```
Fri Oct 10 19:47:40 EDT 2014: runmmfs starting
Removing old /var/adm/ras/mmfs.log.* files:
Loading kernel extension from /usr/lpp/mmfs/bin . . .
/usr/lpp/mmfs/bin/mmfskxload: /usr/lpp/mmfs/bin/mmfs is already loaded at 1355825152.
mmtrace: /tmp/mmfs is low on free space. Required space (estimate): 183M
Fri Oct 10 19:47:47.350 2014: [I] mmfsd initializing. {Version: 4.1.0.4   Built: Oct  8 2014 10:46:32} ...
Fri Oct 10 19:47:47.376 2014: [I] Cleaning old shared memory ...
Fri Oct 10 19:47:47.432 2014: [I] First pass parsing mmfs.cfg ...
Fri Oct 10 19:47:47.456 2014: [I] Enabled automated deadlock detection.
Fri Oct 10 19:47:47.483 2014: [I] Enabled automated deadlock debug data collection.
Fri Oct 10 19:47:47.539 2014: [I] Initializing the main process ...
Fri Oct 10 19:47:47.584 2014: [I] Second pass parsing mmfs.cfg ...
Fri Oct 10 19:47:47.607 2014: [I] Initializing the page pool ...
Fri Oct 10 19:47:47.908 2014: [I] Initializing the mailbox message system ...
Fri Oct 10 19:47:47.963 2014: [I] Initializing encryption ...
Fri Oct 10 19:47:47.994 2014: [I] Encryption: loaded crypto library: IBM CryptoLite for C v4.10.1.5600
(c4T3/GPFSAIX64).
Fri Oct 10 19:47:48.022 2014: [I] Initializing the thread system ...
Fri Oct 10 19:47:48.073 2014: [I] Creating threads ...
Fri Oct 10 19:47:48.138 2014: [I] Initializing inter-node communication ...
Fri Oct 10 19:47:48.199 2014: [I] Creating the main SDR server object ...
Fri Oct 10 19:47:48.233 2014: [I] Initializing the sdrServ library ...
Fri Oct 10 19:47:48.277 2014: [I] Initializing the ccrServ library ...
Fri Oct 10 19:47:48.372 2014: [I] Initializing the cluster manager ...
Fri Oct 10 19:47:50.525 2014: [I] Initializing the token manager ...
Fri Oct 10 19:47:50.617 2014: [I] Initializing network shared disks ...
Fri Oct 10 19:47:53.899 2014: [I] Start the ccrServ ...
Fri Oct 10 19:47:54.412 2014: [N] Connecting to 172.16.20.154 fsaix2 <c0p3>
Fri Oct 10 19:47:54.424 2014: [I] Connected to 172.16.20.154 fsaix2 <c0p3>
Fri Oct 10 19:47:54.441 2014: [N] Connecting to 172.16.20.155 fsaix3 <c0p1>
Fri Oct 10 19:47:54.452 2014: [I] Connected to 172.16.20.155 fsaix3 <c0p1>
Fri Oct 10 19:47:54.468 2014: [N] Connecting to 172.16.20.156 fsaix4 <c0p2>
Fri Oct 10 19:47:54.512 2014: [I] Node 172.16.20.153 (fsaix1) is now the Group Leader.
Fri Oct 10 19:47:54.545 2014: [N] This node (172.16.20.153 (fsaix1)) is now Cluster Manager for tsmgpfs.fsaix1.
Fri Oct 10 19:47:54.974 2014: [N] mmfsd ready
Fri Oct 10 19:47:55 EDT 2014: mmcommon mmfsup invoked. Parameters: 172.16.20.153 172.16.20.153 all
Fri Oct 10 19:47:58 EDT 2014: mounting /dev/gpfs1
Fri Oct 10 19:47:58.179 2014: [I] Command: mount gpfs1 19136646
Fri Oct 10 19:47:58.301 2014: [I] Accepted and connected to 172.16.20.156 fsaix4 <c0p2>
Fri Oct 10 19:47:59.020 2014: [N] Node 172.16.20.153 (fsaix1) appointed as manager for gpfs4.
Fri Oct 10 19:48:00.531 2014: [N] Node 172.16.20.154 (fsaix2) appointed as manager for gpfs1.
Fri Oct 10 19:48:02.389 2014: [N] Node 172.16.20.153 (fsaix1) appointed as manager for gpfs2.
Fri Oct 10 19:48:04.498 2014: [I] Node 172.16.20.153 (fsaix1) completed take over for gpfs4.
Fri Oct 10 19:48:04.611 2014: [N] Node 172.16.20.154 (fsaix2) appointed as manager for gpfs3.
Fri Oct 10 19:48:06.987 2014: [I] Node 172.16.20.153 (fsaix1) completed take over for gpfs2.
Fri Oct 10 19:48:08.715 2014: [I] Node 172.16.20.154 (fsaix2) completed take over for gpfs1.
Fri Oct 10 19:48:12.292 2014: [I] Node 172.16.20.154 (fsaix2) completed take over for gpfs3.
Fri Oct 10 19:48:12.444 2014: [N] Node 172.16.20.153 (fsaix1) appointed as manager for wpargpfs01.
Fri Oct 10 19:48:15.194 2014: [I] Command: successful mount gpfs1 19136646
Fri Oct 10 19:48:15 EDT 2014: finished mounting /dev/gpfs1
Fri Oct 10 19:48:16 EDT 2014: mounting /dev/gpfs2
Fri Oct 10 19:48:16.181 2014: [I] Command: mount gpfs2 18874506
Fri Oct 10 19:48:16.252 2014: [I] Command: successful mount gpfs2 18874506
Fri Oct 10 19:48:16 EDT 2014: finished mounting /dev/gpfs2
```

```
Fri Oct 10 19:48:17 EDT 2014: mounting /dev/gpfs3
Fri Oct 10 19:48:17.398 2014: [I] Command: mount gpfs3 16843006
Fri Oct 10 19:48:17.547 2014: [I] Command: successful mount gpfs3 16843006
Fri Oct 10 19:48:18 EDT 2014: finished mounting /dev/gpfs3
Fri Oct 10 19:48:18 EDT 2014: mounting /dev/gpfs4
Fri Oct 10 19:48:18.598 2014: [I] Command: mount gpfs4 16842754
Fri Oct 10 19:48:18.705 2014: [I] Command: successful mount gpfs4 16842754
Fri Oct 10 19:48:19 EDT 2014: finished mounting /dev/gpfs4
Fri Oct 10 19:48:19.222 2014: [I] Node 172.16.20.153 (fsaix1) completed take over for wpargpfs01.
Fri Oct 10 19:48:19 EDT 2014: mounting /dev/wpargpfs01
Fri Oct 10 19:48:19.598 2014: [I] Command: mount wpargpfs01 15859840
Fri Oct 10 19:48:19.683 2014: [I] Command: successful mount wpargpfs01 15859840
Fri Oct 10 19:48:20 EDT 2014: finished mounting /dev/wpargpfs01
Fri Oct 10 21:16:05 EDT 2014: mmremote: unmountFileSystems gpfs1  ...
Fri Oct 10 21:16:06.240 2014: [I] Command: unmount gpfs1
Fri Oct 10 21:16:06.261 2014: [I] Command: successful unmount gpfs1
Fri Oct 10 21:16:45.066 2014: [N] Node 172.16.20.154 (fsaix2) resigned as manager for gpfs1.
Fri Oct 10 21:16:45.075 2014: File system has been deleted.
Fri Oct 10 21:19:03 EDT 2014: mmremote: unmountFileSystems all -f ...
Fri Oct 10 21:19:04.143 2014: [I] Command: unmount wpargpfs01 forced
Fri Oct 10 21:19:04.154 2014: [I] Command: successful unmount wpargpfs01
Fri Oct 10 21:19:04.185 2014: [I] Command: unmount gpfs4 forced
Fri Oct 10 21:19:04.202 2014: [I] Command: successful unmount gpfs4
Fri Oct 10 21:19:04.232 2014: [I] Command: unmount gpfs3 forced
Fri Oct 10 21:19:04.254 2014: [I] Command: successful unmount gpfs3
Fri Oct 10 21:19:04.285 2014: [I] Command: unmount gpfs2 forced
Fri Oct 10 21:19:04.302 2014: [I] Command: successful unmount gpfs2
Fri Oct 10 21:19:08 EDT 2014: mmremote: Initiating GPFS shutdown ...
Fri Oct 10 21:19:10.257 2014: [N] Restarting mmsdrserv
Fri Oct 10 21:19:10.801 2014: [N] mmfsd is shutting down.
Fri Oct 10 21:19:10.810 2014: [N] Reason for shutdown: Normal shutdown
Fri Oct 10 21:19:11 EDT 2014: mmcommon mmfsdown invoked.  Subsystem: mmfs Status: down
Fri Oct 10 21:19:11 EDT 2014: mmcommon: Unmounting file systems ...
Fri Oct 10 21:19:15 EDT 2014: mmremote: Starting the mmsdrserv daemon ...
Fri Oct 10 21:19:15 EDT 2014: mmremote: Completing GPFS shutdown ...
```

Configuration attributes

Spectrum Scale generates some critical log messages that are always sent to the system logging service (syslog on Linux and AIX, AIX errlog, and Windows Event Viewer) and Spectrum Scale log file. Starting with Spectrum Scale 4.1, many Spectrum Scale messages can be additionally sent to syslog (it applies to syslog only). The following attributes are used to control the Spectrum Scale messages:

systemLogLevel

This configuration attribute specifies the minimum severity level for messages sent to the system log. The severity levels from highest to lowest priority are: alert, critical, error, warning, notice, configuration, informational, detail, and debug. The default value is `notice`. The value specified for this attribute can be any severity level, or the value `none` can be specified so no messages are sent to the system log. This attribute only affects messages originating in the Spectrum Scale daemon (**mmfsd**).

mmfsLogLevel

This option specifies the minimum severity level for messages sent to the Spectrum Scale log file (`/var/adm/ras/mmfs.log.latest`) or `none` for no messages. It uses the same message severity levels used by the **systemLogLevel** attribute and the default level is `detail`.

Both **systemLogLevel** and **mmfsLogLevel** can be changed by the **`mmchconfig`** command and are independent. For more information, see the **`mmchconfig`** command in GPFS V4.1: Administration and Programming Reference, SA23-1452-00.

Creating a Spectrum Scale dump directory

When Spectrum Scale encounters an internal problem, certain state information is saved in the Spectrum Scale dump directory for later analysis by IBM service.

One suggestion is that you create a directory for the placement of the problem determination information. Do not place the information in a Spectrum Scale file system because that information might not be available if Spectrum Scale fails.

The default location of the dump directory is `/tmp/mmfs`. Administrators who want to use a separate directory for Spectrum Scale dumps can change the directory by issuing the following command **`/usr/lpp/mmfs/bin/mmchconfig dataStructureDump=/full directory path`**.

Creating a master Spectrum Scale log file

Since the Spectrum Scale is a file system that runs on multiple nodes, the Spectrum Scale log can show problems on one node that were originated from another node.

It is suggested to merge the Spectrum Scale logs in pursuit of a problem. Having accurate time stamps aid the analysis of the sequence of events.

Before following any of the debug steps, consider some suggested practices:

1. Synchronize all clocks of all nodes in the Spectrum Scale cluster. If this is not done, and clocks on different nodes are out of sync, there is no way to establish the real timeline of events occurring on multiple nodes. Therefore, a merged error log is less useful for determining the origin of a problem and tracking its effects.
2. Merge and chronologically sort all of the Spectrum Scale log entries from each node in the cluster. The **`--gather-logs`** option of the **`gpfs.snap`** command as follows:

   ```
   /usr/lpp/mmfs/bin/gpfs.snap --gather-logs -d /tmp/logs -N all
   ```

The system displays information similar to:

```
gpfs.snap: Gathering mmfs logs ...
gpfs.snap: The sorted and unsorted mmfs.log files are in /tmp/logs
```

If the **`--gather-logs`** option is not available on your system, you can create your own script to achieve the same task; use **`/usr/lpp/mmfs/samples/gatherlogs.sample.sh`** as an example.

When **`gpfs.snap`** is issued with **`--gather-logs`** it only merges the logs, it is not intended to replace the **`gpfs.snap`** command.

8.6.4 The gpfs.snap command

The **`gpfs.snap`** command creates an informational system snapshot at a single point in time. This system snapshot consists of cluster configuration, disk configuration, network configuration, network status, Spectrum Scale logs, dumps, and traces. Use the **`gpfs.snap`**

command as one of the main tools to gather preliminary data when a Spectrum Scale problem is encountered, such as a hung file system, a hung Spectrum Scale command, or an *mmfsd* daemon assert.

The information gathered with the **gpfs.snap** command can be used with other information (for example, Spectrum Scale internal dumps, traces, and kernel thread dumps) to solve a Spectrum Scale problem. The syntax of the **gpfs.snap** command is as follows:

```
gpfs.snap [-c "CommandString"] [-d OutputDirectory] [-m | -z] [-a | -N
{Node[,Node...] | NodeFile | NodeClass}][--check-space | --no-check-space |
--check-space-only][--deadlock [--quick]] [--exclude-aix-disk-attr]
[--exclude-aix-lvm] [--exclude-net] [--exclude-merge-logs] [--gather-logs]
[--mmdf] [--prefix]
```

These options are used with **gpfs.snap**:

-c "CommandString"

Specifies the command string to run on the specified nodes. When this option is specified, the data collected is limited to the result of the specified command string. The standard data collected by **gpfs.snap** is not collected. CommandString can consist of multiple commands, which are separated by semicolons (;) and enclosed in double quotation marks (").

-d OutputDirectory

Specifies the output directory. The default is `/tmp/gpfs.snapOut`.

-m

Specifying this option is equivalent to specifying `--exclude-merge-logs` with `-N`.

-z

Collects gpfs.snap data only from the node on which the command is invoked. No master data is collected.

-a

Directs gpfs.snap to collect data from all nodes in the cluster. This is the default.

-N {Node[,Node ...] | NodeFile | NodeClass}

Specifies the nodes from which to collect gpfs.snap data. This option supports all defined node classes.

--check-space

Specifies that space checking is performed before collecting data.

--no-check-space

Specifies that no space checking is performed. This is the default.

---check-space-only

Specifies that only space checking is performed. No data is collected.

--deadlock

Collects only the minimum amount of data necessary to debug a deadlock problem. Part of the data collected is the output of the mmfsadm dump all command. This option ignores all other options except for **-a**, **-N**, **-d**, and **--prefix**.

--quick

Collects less data when specified along with the **--deadlock option**. The output includes **mmfsadm dump most**, **mmfsadm dump kthreads**, and 10 seconds of trace in addition to the usual **gpfs.snap** output.

--exclude-aix-disk-attr

Specifies that data about AIX disk attributes is not collected. Collecting data about AIX disk attributes on an AIX node that has many disks could be time-consuming, so using this option could help improve performance.

--exclude-aix-lvm

Specifies that data about the AIX Logical Volume Manager (LVM) is not collected.

--exclude-net

Specifies that network-related information is not collected.

--exclude-merge-logs

Specifies that merge logs and waiters are not collected.

--gather-logs

Gathers, merges, and chronologically sorts all of the `mmfs.log` files. The results are stored in the directory specified with the **-d** option.

--mmdf

Specifies that mmdf output is collected.

--prefix

Specifies that the prefix name gpfs.snap is added to the tar file.

Use the **-z** option to generate a non-master snapshot. This is useful if there are many nodes on which to take a snapshot, and only one master snapshot is needed. For a Spectrum Scale

problem within a large cluster (hundreds or thousands of nodes), one strategy might call for a single master snapshot (one invocation of **gpfs.snap** with no options), and multiple non-master snapshots (multiple invocations of **gpfs.snap** with the **-z** option).

Use the **-N** option to obtain **gpfs.snap** data from multiple nodes in the cluster. When the **-N** option is used, the **gpfs.snap** command takes non-master snapshots of all the nodes specified with this option and a master snapshot of the node on which it was invoked.

8.6.5 Using the gpfs.snap command

The **gpfs.snap** script gathers all the general documentation that is needed for IBM Service to investigate most issues. The script creates a `.tar` file in the `/tmp/gpfs.snapOut` location. Depending on the options that are used to invoke it, the amount and scope of the documentation that is gathered can vary. The **gpfs.snap** command creates a *master* snapshot from the node where the command was executed and *non-master* snapshots from all nodes of the clusters. The difference between a master snapshot and a non-master snapshot is the data that is gathered. A master snapshot contains all data that a non-master snapshot has and some additional information.

Usually in a gpfs.snap master snapshot, we see the following information:

- The output of these commands:
  - `mmauth`
  - `mmgetstate -a`
  - `mmlscluster`
  - `mmlsconfig`
  - `mmlsdisk`
  - `mmlsfileset`
  - `mmlsfs`
  - `mmlspolicy`
  - `mmlsmgr`
  - `mmlsnode -a`
  - `mmlsnsd`
  - `mmlssnapshot`
  - `mmremotecluster`
  - `mmremotefs`
  - `tsstatus`
  - The contents of the `/var/adm/ras/mmfs.log.*` file (on all nodes in the cluster)

Because the size of the cluster is not known, taking one master snapshot from one node and a non-master snapshot from all the other nodes is necessary (**gpfs.snap** without any flags). The node that is chosen on which to run the master snap must be able to communicate (with the **ssh** or **rsh**) to all other nodes in the cluster.

When the **gpfs.snap** command is issued, it gathers information from the specific operating system and platform and the Spectrum Scale specific information.

When executing the **gpfs.snap**, the output should be similar to Example 8-21.

Example 8-21 The gpfs.snap output from an AIX Spectrum Scale node

```
root@fsaix2:/>/usr/lpp/mmfs/bin/gpfs.snap
gpfs.snap started at Mon Oct 13 18:21:18 EDT 2014.
Gathering common data.
Gathering AIX specific data.
Gathering trace reports and internal dumps...
gpfs.snap:  Spawning remote gpfs.snap calls. Master is fsaix2.
```

```
This may take a while.

Copying file /tmp/gpfs.snapOut/15204378/gpfs.snap.fsaix3_1013182330.out.tar.gz from fsaix3
...
Successfully copied file /tmp/gpfs.snapOut/15204378/gpfs.snap.fsaix3_1013182330.out.tar.gz
from fsaix3.

Copying file /tmp/gpfs.snapOut/15204378/gpfs.snap.fsaix1_1013182330.out.tar.gz from fsaix1
...
Successfully copied file /tmp/gpfs.snapOut/15204378/gpfs.snap.fsaix1_1013182330.out.tar.gz
from fsaix1.
Packaging master node data.
/tmp/gpfs.snapOut/15204378/collect/gpfs.snap.fsaix2_master_1013182118.out.tar.gz
Packaging all data.
gpfs.snap completed at Mon Oct 13 18:24:03 EDT 2014
###############################################################################
Send file /tmp/gpfs.snapOut/15204378/all.1013182118.tar to IBM Service
###############################################################################
```

In Example 8-22, the **gpfs.snap** command created a single file named /tmp/gpfs.snapOut/15204378/all.1013182118.tar. The output file is created based on the timestamp that the gpfs.snap is issued and contains the files as shown in Example 8-22.

Example 8-22 Listing gpfs.snap output

```
root@fsaix2:/tmp/gpfs.snapOut/15204378>tar -tvf all.1013182118.tar
/gpfs.snap.fsaix1_1013182330.out.tar.gz
/gpfs.snap.fsaix2_master_1013182118.out.tar.gz
/gpfs.snap.fsaix3_1013182330.out.tar.gz
/remote.gpfs.snap.output_1013182118
```

In this snapshot, a file named gpfs.snap.fsaix2_master_1013182118.out.tar.gz contains the master snapshot because the command was issued on the *fsaix2* Spectrum Scale node and non-master snapshots from *fsaix1* and *fsaix3* nodes.

If the node on which the **gpfs.snap** command is run is not a file system manager node, **gpfs.snap** creates a non-master snapshot on the file system manager nodes.

To illustrate the information that is usually obtained from a **gpfs.snap** command, the following sections are created:

- “General gpfs.snap contents for all platforms” on page 437
- “Collecting gpfs.snap data on AIX” on page 438
- “Collecting gpfs.snap data on Linux” on page 438
- “Collecting gpfs.snap data on Windows operating systems” on page 439

General gpfs.snap contents for all platforms

Some Spectrum Scale-related information is always obtained by the **gpfs.snap** command, regardless of the platform:

- The output of these commands:
  - ls -l /user/lpp/mmfs/bin
  - mmdevdiscover
  - tspreparedisk -S
  - mmfsadm dump malloc
  - mmfsadm dump fs
  - df -k

  - `ifconfig` *interface*
  - `ipcs -a`
  - `ls -l /dev`
  - `mmfsadm dump alloc hist`
  - `mmfsadm dump alloc stats`
  - `mmfsadm dump allocmgr`
  - `mmfsadm dump allocmgr hist`
  - `mmfsadm dump allocmgr stats`
  - `mmfsadm dump cfgmgr`
  - `mmfsadm dump config`
  - `mmfsadm dump dealloc stats`
  - `mmfsadm dump disk`
  - `mmfsadm dump mmap`
  - `mmfsadm dump mutex`
  - `mmfsadm dump nsd`
  - `mmfsadm dump rpc`
  - `mmfsadm dump sgmgr`
  - `mmfsadm dump stripe`
  - `mmfsadm dump tscomm`
  - `mmfsadm dump version`
  - `mmfsadm dump waiters`
  - `netstat` with the -i, -r, -rn, -s, and -v options
  - `ps -edf`
  - `vmstat`
- The contents of these files:
  - `/etc/syslog.conf or /etc/syslog-ng.conf`
  - `/tmp/mmfs/internal*`
  - `/tmp/mmfs/trcrpt*`
  - `/var/adm/ras/mmfs.log.*`
  - `/var/mmfs/gen/*`
  - `/var/mmfs/etc/*`
  - `/var/mmfs/tmp/*`
  - `/var/mmfs/ssl/*` except for `complete.map` and `id_rsa` files

Collecting gpfs.snap data on AIX

These items are always obtained by the **`gpfs.snap`** command when gathering data for an AIX node:

- The output of these commands:
  - `errpt -a`
  - `lssrc -a`
  - `lslpp -hac`
  - `no -a`
- The contents of these files:
  - `/etc/filesystems`
  - `/etc/trcfmt`

Collecting gpfs.snap data on Linux

These items are always obtained by the **`gpfs.snap`** command when gathering data for a Linux node:

- The output of these commands:
  - `dmesg`

  - `fdisk -l`
  - `lsmod`
  - `lspci`
  - `rpm -qa`
  - `rpm --verify gpfs.base`
  - `rpm --verify gpfs.docs`
  - `rpm --verify gpfs.gpl`
  - `rpm --verify gpfs.msg.en_US`
- The contents of these files:
  - `/etc/filesystems`
  - `/etc/fstab`
  - `/etc/*release`
  - `/proc/cpuinfo`
  - `/proc/version`
  - `/usr/lpp/mmfs/src/config/site.mcr`
  - `/var/log/messages*`

Collecting gpfs.snap data on Windows operating systems

These items are always obtained by the **`gpfs.snap`** command when gathering data for a Windows operating system node:

- The output from **`systeminfo.exe`**
- Any raw trace files `*.tmf` and `mmfs.trc*`
- The `*.pdb` symbols from `/usr/lpp/mmfs/bin/symbols`

8.7 Collecting IBM Spectrum Scale debug information

This section describes some commands that are used in collecting debugging information for the Spectrum Scale cluster and some of its major components. It describes some of the commands that a system administrator can use when attempting to identify the root cause of a problem and to generate data to send to IBM Service Center when requested.

8.7.1 Collecting specific information

When debugging some Spectrum Scale problems the system administrators might need to gather more detailed information. In this topic, we show some of the most used commands to collect specific data. Sometimes instead of taking a full gpfs.snap the following commands can be used:

- `mmdiag`
- `mmfsadm`

mmdiag command

This command gives the system administrator an interface to query the mmfs daemon and gather useful information such as memory pool usage, I/O history, pending *Remote Procedure Calls* (RPCs), current trace levels, and more without the potential of causing an interruption in the file systems' or clusters' availability.

The options to the **`mmdiag`** command, as shown in Example 8-23 on page 440, simplify the output of the **`mmfsadm dump`** options that are used in previous versions of Spectrum Scale and can be incorporated in scripts that are used to monitor cluster status.

Example 8-23 Options for the mmdiag command

```
mmdiag usage:
  --help|-h      Display this help message
  --all          Display everything
  --version      Display information about the running GPFS build
  --waiters      Display mmfsd threads waiting for events
  --deadlock     Display waiters exceeding deadlock detection thresholds
  --threads      Display mmfsd thread stats and the list of active threads
  --memory       Display information about mmfsd memory usage
  --network      Display information about mmfsd network connections
  --config       Display configuration parameters and their settings
  --trace        Display current trace status and trace levels
  --assert       Display current dynamic assert status and levels
  --iohist       Display recent IO history
  --tokenmgr     Display information about token management
  --commands     Display list of running or waiting commands
  --dmapi [session|event|token|disposition|all]
                 Display various DMAPI information
  --rpc [node[=name]|size|message|all|nn{S|s|M|m|H|h|D|d}]
                 Display RPC performance statistics
  --stats        Display some general GPFS stats
```

mmdiag --rpc

One of the problems that can directly affect the Spectrum Scale performance is the network. Since the Spectrum Scale uses RPCs to communicate between the nodes, the **mmdiag** command can be used to measure RPC point-to-point calls. The RPC latency is defined as the difference between the RPC round-trip time, as measured on the sending node, and the RPC execution time, as measured on the target node. The **mmdiag** command was improved to provide more detailed information about RPC execution times. To diagnose specific RPC problems, the **mmdiag** command can be used with the **--rpc** option to display the cached RPC-related statistics. The command usage can be as follows:

```
mmdiag --rpc [node[=hostname]|size|message|all|nn{S|s|M|m|H|h|D|d}]
```

Following are the valid options:

(none)	Per node RPC latency statistics (TCP, verbs, and mixed)
node[=name]	All per node stats (channel wait, send time TCP, send time verbs, receive time TCP, latency TCP, latency verbs, and latency mixed).
size	Per size range statistics
message	Per message type RPC execution time
all	Every message for all nodes
nn	**S|s|M|m|H|h|D|d** Per node RPC latency statistics for the latest number of intervals, which are specified by nn, for the interval specified by one of the characters: **S,s**: Display second intervals only **M,m**: Displays first the second intervals since the last-minute boundary followed by minute intervals **H,h**: Display sec/min intervals since their last min/hour boundary, then hours **D,d**: display sec/min/hour intervals since their last min/hour/day boundary, then days

When issuing the **mmdiag** --rpc 10s command, the output looks like what is shown in Example 8-24 on page 441.

Example 8-24 mmdiag --rpc 10s output

```
root@fsaix1:/>mmdiag --rpc 10s

=== mmdiag: RPC statistics ===

RPC (msec) aggregated statistics for node fsaix3
TCP RPC Latency
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
Verbs RPC Latency
Mixed RPC Latency

RPC (msec) aggregated statistics for node fsaix2
TCP RPC Latency
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
sec:  average =           0.000, min =           0.000, max =           0.000, count =        0
Verbs RPC Latency
Mixed RPC Latency
```

Note: The command was issued on node *fsaix1* but no output from this node was given. This is because this command measures the latency between the local node and the other Spectrum Scale nodes over the network. For this command, the local node name is not shown.

When executed without any parameters, the **mmdiag** command shows only basic information about the Spectrum Scale version and uptime as shown in Example 8-25.

Example 8-25 The mmdiag output

```
root@fsaix1:/var/adm/ras>/usr/lpp/mmfs/bin/mmdiag
Current GPFS build: "4.1.0.4 ".
Built on Oct  8 2014 at 10:46:32
Running 5 hours 2 minutes 59 secs
Run mmdiag --help for more options
```

Usually, the rpc statistics are analyzed by the IBM Service Center.

mmfsadm command

The **mmfsadm** command and several of its options can be important during initial problem determination and when gathering documentation for submission to IBM Service. It is used to query the mmfs daemon without the use of locking. The **mmfsadm dump** command was

improved in Spectrum Scale version 4.1 to avoid a possible core dump on the mmfsd daemon under some specific circumstances.

Attention: It is highly recommended to only use the **mmfsadm** command under the direction of your IBM service representative.

The **mmfsadm** command is also used to show information about *waiters*.

Following are some of the most common options that can be used for the command:

cleanup Delete shared segments that are left by a previously failed GPFS daemon without restarting the daemon.

dump what Dumps the state of many internal state values that might be useful in determining the sequence of events. The **what** parameter can be set to all, indicating that all available data should be collected, or to another value, indicating more restricted collection of data. The output is presented to STDOUT and should be collected by redirecting STDOUT.

showtrace Shows the current level for each subclass of tracing available in GPFS. Trace level 14 provides the highest level of tracing for the class and trace level 0 provides no tracing. Intermediate values exist for most classes. More tracing requires more storage and results in a higher probability of overlaying the required event.

trace *class n* Sets the trace class to the value specified by *n*. Actual trace gathering only occurs when the **mmtracectl** command has been issued.

Output from the **mmfsadm** command is required in almost all cases where a Spectrum Scale problem is being reported. The **mmfsadm** command collects data only on the node where it is issued. Depending on the nature of the problem, the output might be required from several or all nodes. The **mmfsadm** output from the file system manager is often required.

To determine where the file system manager is, issue the **mmlsmgr** command as shown in Example 8-26.

Example 8-26 The mmlsmgr output

```
root@fsaix1:/var/adm/ras>/usr/lpp/mmfs/bin/mmlsmgr
file system      manager node
---------------- ------------------
gpfs3            172.16.20.153 (fsaix1)
wpargpfs01       172.16.20.153 (fsaix1)
gpfs2            172.16.20.154 (fsaix2)
gpfs4            172.16.20.154 (fsaix2)
Cluster manager node: 172.16.20.153 (fsaix1)
```

mmfsadm dump waiters command

Waiters are a normal part of workload on a Spectrum Scale cluster. They represent work that the mmfs daemon must perform at the time the **mmfsadm dump waiters** command was executed. Under normal conditions, waiters seen in one interaction of the command have been completed before you run it again. However, for example, if the Spectrum Scale file system is hanging or performance is slow, the same waiters gain time upon multiple queries.

The ability to gather waiters from the cluster and make decisions based on those waiters is an important step in determining what might be happening on the cluster.

If you have a few nodes and no remote clusters, you can gather waiters from all of the nodes.

Gathering data from hundreds or thousands of nodes can be difficult as the network plays a larger role and can cause command timeouts and failures. However, to get a clear picture of what the problem might be, this approach is necessary.

If file systems are remotely mounted, all remote nodes in any remote cluster are part of the local cluster and the commands that are used to gather data on your local cluster must be gathered on those remote clusters also. This approach means logging in to one of the nodes in the remote cluster and issuing the same command on those nodes as in the local cluster. If access to the remote cluster is not possible, instead gather the data that can be gathered locally for analysis.

Gathering the waiters

To understand where the problem might be, waiters must be gathered from all nodes in the cluster: All nodes in the local cluster, and if applicable, nodes in remote clusters.

The **mmfsadm dump waiters** command can be executed from any node to identify if the system is experiencing long waiters as shown in Example 8-27.

Example 8-27 Command to gather waiters

```
/usr/lpp/mmfs/bin/mmdsh -F [all | wcoll-file] mmfsadm dump waiters >
/tmp/mmfs/allwaiters.$(date +\%m\%d\%H\%M)
```

This command can be executed a number of times to show movement of waiters, which might help in determining the cause of current issue. See “If the kernel extension cannot be unloaded, rebooting is mandatory.” on page 468 for the procedure to analyze gathered waiters.

mmfsadm dump all

The **dump all** option of the **mmfsadm command** is used when detailed information about the internal state of the daemon is needed for problem determination. The nodes on which the **dump all** option is run is usually determined by the nodes that have waiters at the time a problem is occurring. See Example 8-28.

Example 8-28 Command to gather detailed mmfs daemon state

```
/usr/lpp/mmfs/bin/mmfsadm dump all > /tmp/$(hostname -s).dumpall
```

mmfsadm dump kthreads

Sometimes, you have to determine what the mmfsadm kernel threads are doing. The **dump kthreads** option as shown in Example 8-29, lists all the kernel threads that are running at the time the command is executed. Use this output if a kernel thread does not appear to be responding. The thread and its associated traceback is displayed.

This output can be very large and time consuming to collect, and as a result, is not part of the general information that is requested when a reported problem occurs.

Example 8-29 Command to gather mmfs kern threads

```
/usr/lpp/mmfs/bin/mmfsadm dump kthreads > $(hostname -s).dumpkthreads
```

mmfsadm test nsd qcalc

The **mmfsadm test nsd qcalc** command can be used to identify if the amount of buffer space that is used for the NSDs has not exceeded the default value.

The output from this command should be similar to Example 8-30.

Example 8-30 mmfsadm test nsd qcalc output

```
root@fsaix2:/tmp/gpfs.snapOut/15204378>mmfsadm test nsd qcalc

NSD calculation uses these simluated values:
    Name                          Value      User override
  IsNSDRaidConfigured                 0
  PagePoolSize               1073741824
  nsdBufSpace                        30
  maxBlockSize                 1048576
  nsdThreadMethod                     1
  nsdMaxWorkerThreads               512
  nsdMinWorkerThreads                16

  nsdMultiQueue                     256
  nsdSmallBufferSize              65536
  nsdSmallThreadRatio                 7
  nsdThreadsPerDisk                   3
  nsdThreadsPerQueue                  3
  numNSDServerDisks                   0

  nsdRAIDThreadMethod                 0
  nsdRAIDAllowTraditionalNSD          0
  nsdRAIDDesiredThreadPct           100
  nsdRAIDMultiQueue                  -1
  nsdRAIDSmallBufferSize             -1
  nsdRAIDSmallThreadRatio            -1
  nsdRAIDThreadsPerVDisk             -1
  nsdRAIDThreadsPerQueue             -1
  numGNRServerDisks                   0

GNR calculation disabled because:
  IsNSDRaidConfigured is zero
  numGNRServerDisks is zero

Traditional NSD settings:
 Config data for NsdQueueTraditional (NsdQueueUseNsdIO):
 Input config parms:
  nsdThreadsPerDisk: 3, nsdMinWorkerThreads: 16, nsdMaxWorkerThreads 512,
nsdThreadsPerQueue 3, numServerDisksForType 0, nsdThreadMethod 1
  nsdSmallBufferSize 65536, nsdBigBufferSize: 0, nsdSmallThreadRatio 7, maxNumNsdQueues 256
 Derived config parms:
  threadRatio: 7, threadsPerQueue: 3, numLargeQueues 21, numSmallQueues: 149
  largeBufferSize: 1048576, smallBufferSize: 65536, desiredThreadsForType 16
 Thread Summary
  Large Threads 63, Small Threads 447
  Large Buff Size 66060288, Small Buff Size 29294592, Total Buff Size 95354880

GNR configured disabled - no calculation will be done for GNR.

desiredThreads                16 (based solely on disk count)
desiredThreadsForNSD          16 (based solely on disk count)
actual NSD threads           510 total:     447 small      63 large
```

```
Error code                    0

Available (Simulated) NSD Buffer Space    322122540
Used      (Simulated) NSD Buffer Space     95354880
Percent used of NSD Buffer Space              29.6 %
```

In this example, the percent buffer space used is 29.6%, which indicates the default value is enough. The following NSD parameters are associated with thread tuning:

nsdThreadsPerDisk Is a parameter from previous versions of GPFS and the default value is 3. It represents an estimated value on how many threads can keep a single NSD LUN busy without inducing a detrimental amount of backlog. With high-bandwidth NSDs like disks, this number may be increased.

nsdBufSpace Starting in GPFS 3.5, nsdBufSpace places an indirect maximum limit on the number of NSD threads at start time, by limiting the available space for the buffers dedicated to NSD threads.

Note: Before changing any Spectrum Scale parameter that might affect the performance of your system, contact your IBM Service Representative.

8.8 Working with IBM Spectrum Scale trace facility

The Spectrum Scale tracing uses various tracing facilities based on the operating system. In AIX, it is based on the AIX trace facility. In Linux, it uses Spectrum Scale specific trace subsystem. And on Windows operating system, it uses the Windows ETL subsystem.

In Spectrum Scale version 4.1, the tracing facility on Linux systems was improved to reduce the overhead that was caused on Linux environments when the Spectrum Scale trace was enabled. This new improvement can allow the tracing options to be enabled by default in many Linux environments.

The amount of detail that the traces produce varies, depending on the trace levels that are set. You can use the **mmdiag --trace** or **mmfsadm showtrace** commands to check the current levels.

The **mmtracectl** command sets up and enables tracing using default settings for various common problem situations. Using this command improves the probability of gathering accurate and reliable problem determination information.

Attention: It is recommended to use this command only under the direction of your IBM service representative.

The **mmtracectl** command is best used when tracing must span a recycle operation, whether it is daemon or node. There are more options with **mmtracectl** that allow you to take traces on multiple nodes, based on the actions of one. See Example 8-31 for syntax for **mmtracectl**.

Example 8-31 The mmtracectl syntax usage

```
mmtracectl {--start | --stop | --off | --set}
             [--trace={io | all | def | "Class Level [Class Level ...]"}]
             [--trace-recycle={off | local | global | globalOnShutdown}]
```

```
[--aix-trace-buffer-size=BufferSize]
[--tracedev-buffer-size=BufferSize]
[--trace-file-size=FileSize] [--trace-dispatch={yes | no}]
[--tracedev-compression-level=Level]
[--tracedev-write-mode={blocking | overwrite}]
[--tracedev-overwrite-buffer-size=Size]
[--format | --noformat]
[-N {Node[,Node...] | NodeFile | NodeClass}]
```

The **mmtracectl** command can also be used to perform the following functions:

- Start or stop tracing.
- Turn tracing on (start or set trace recycle) or off on the next session. This is a persistent setting to automatically start trace each time Spectrum Scale starts.
- Allow for predefined trace levels: io, all, and def, as well as user-specified trace levels.
- Allow for changing the size of trace buffer sizes for AIX and all others using the **tracedev** option.
- Trace recycle functions, which allow for never cycling traces (off option), cycling traces on all nodes when Spectrum Scale ends abnormally (global option), and cycling traces any time Spectrum Scale goes down on all nodes (globalOnShutdown option).
- For Linux nodes only, this command allows you to change:
  - The trace writing mode
  - The raw data compression level

8.8.1 Generating Spectrum Scale tracing information

When asked by your IBM service representative, you can configure and use the trace properly by following these steps:

1. Issue the **mmlsconfig dataStructureDump** command to verify that a directory for dumps was created when the cluster was configured. Use the **mmtracectl** command as instructed by service personnel to set trace configuration parameters as required if the default parameters are insufficient. If wanted, specify another location as explained in the “Creating a Spectrum Scale dump directory” on page 433.
2. Ensure that the correct trace level is configured. Usually the level 4 is enough to diagnose most of the problems but you may be asked from IBM Service to change to a different level. The trace level can be set to a value 0 - 14, which represents an increasing level of detail. A value of 0 turns tracing off. To display the trace level in use, issue the **mmfsadm showtrace** command as shown in Example 8-32.

Example 8-32 Partial mmfsadm showtrace output

```
=Current trace levels:
       alloc :  0   (disk space allocation)
    allocmgr :  0   (allocation manager)
       basic :  0   ('basic' classes)
         brl :  0   (byte range locks)
     cleanup :  0   (cleanup routines)
         cmd :  0   (TS commands)
      defrag :  0   (defragmentation)
  dentryexit :  0   (daemon routine entry/exit)
        disk :  0   (physical disk I/O)
       dmapi :  0   (data management)
```

```
          ds :  0   (data shipping)
      errlog :  0   (error logging)
          fs :  0   (file system)
        fsck :  0   (online multinode fsck)
(...)
```

3. To change the tracing level to the default “4”, you can execute the **mmtracectl** command as shown in Example 8-33.

Example 8-33 The mmtracectl command to change the default tracing level

```
root@fsaix2:/>mmtracectl --set --trace=def
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
mmfsaroot@fsaix2:/>mmfsadm showtrace
Current trace levels:
       alloc :  4   (disk space allocation)
    allocmgr :  4   (allocation manager)
       basic :  4   ('basic' classes)
         brl :  4   (byte range locks)
     cleanup :  4   (cleanup routines)
         cmd :  4   (TS commands)
      defrag :  4   (defragmentation)
  dentryexit :  4   (daemon routine entry/exit)
        disk :  4   (physical disk I/O)
       dmapi :  4   (data management)
(...)
```

4. To start the tracing facility on all nodes, issue the following command:

   ```
   mmtracectl --start
   ```

5. Re-create the problem.

6. When the event to be captured occurs, stop the trace as soon as possible by issuing this command:

   ```
   mmtracectl --stop
   ```

7. The output of the Spectrum Scale trace facility is stored in `/tmp/mmfs`, unless the location was changed by **mmchconfig**. After the **mmtracectl --stop command** is executed, the output should be similar to Example 8-34.

Example 8-34 The mmtracectl --stop command output

```
fsaix1:  mmtrace: move /tmp/mmfs/trcfile.fsaix1 /tmp/mmfs/trcfile.141013.18.08.08.12451928.fsaix1
fsaix1:  mmtrace: formatting /tmp/mmfs/trcfile.141013.18.08.08.12451928.fsaix1 to  /tmp/mmfs/trcrpt.141013.18.08.08.12451928.fsaix1.gz
fsaix2:  mmtrace: move /tmp/mmfs/trcfile.fsaix2 /tmp/mmfs/trcfile.141013.18.08.08.12714128.fsaix2
fsaix2:  mmtrace: formatting /tmp/mmfs/trcfile.141013.18.08.08.12714128.fsaix2 to /tmp/mmfs/trcrpt.141013.18.08.08.12714128.fsaix2.gz
fsaix3:  mmtrace: move /tmp/mmfs/trcfile.fsaix3 /tmp/mmfs/trcfile.141013.18.08.08.7798958.fsaix3
fsaix3:  mmtrace: formatting /tmp/mmfs/trcfile.141013.18.08.08.7798958.fsaix3 to /tmp/mmfs/trcrpt.141013.18.08.08.7798958.fsaix3.gz
```

   The file named `trcrpt.141013.18.08.08.12451928.<nodename>.gz` should be saved and sent to IBM for further analysis.

8. If the problem results in a shutdown and restart of the Spectrum Scale daemon, set the `traceRecycle` variable with **--trace-recycle** as described in the **mmtracectl** command to ensure that Spectrum Scale traces are performed at the time the error occurs.

If the problem requires more detailed tracing, the IBM Support Center personnel might ask you to modify the Spectrum Scale trace levels. Use the **mmtracectl** command to establish the required trace classes and levels of tracing. The syntax to modify trace classes and levels is as follows:

```
mmtracectl --set --trace={io | all | def | "Class Level [Class Level ...}"]
```

For example, to configure the trace level for I/O, start and stop the trace, the following commands can be executed:

- `mmtracectl --set --trace=io`
- `mmtracectl --start`
- `mmtracectl --stop`

To clear the trace settings, restoring the original value as “0” and ensuring that tracing is turned off, issue:

```
mmtracectl --off
```

8.8.2 Trace data analysis commands and scripts

This section describes common commands and scripts used for analyzing collected data.

The AWK scripts listed in this section are in the `/usr/lpp/mmfs/samples/debugtools` directory. They are used to help analyze the **mmfs trcrpts** (`trsum.awk`) file or the `/var/log/messages` on Linux (`fsstructlx.awk`), or the output from the **errpt -a** command on AIX (`fsstruct.awk`).

trsum.awk

This **.awk** script takes a `trcrpt` data file that has been gathered by using either the **mmtrace** or **mmtracectl** method and divides the trace into stanzas that help you with a visual understanding of what occurred over the time period that the trace was gathered.

The syntax for `trsum.awk` is as follows:

```
trsum.awk [detail=0] trcrpt-file > somefile
```

In the syntax, `detail=0` indicates to omit the detail section of the `trsum` and only generate the histogram sections.

The stanzas are in the usual `trsum` output as shown in Table 8-1.

Table 8-1 The trsum stanza outputs

Stanza	Details shown
Detail trace section	Shows system call entry, exit, and the elapsed time.
Elapsed time	Duration of the trace based on the size of the trace buffers and node activity.
Operations stats	Spectrum Scale operation that is performed and total time, count, and average time to perform those operations.
User thread stats	Per thread account of time broken out into time waiting on Spectrum Scale and the application.
Total App-read/write	Histograms-based application reads/writes information.
Max concurrent App-read/write	Maximum reads/writes that the application had concurrent at any one time.

Stanza	Details shown
Total I/Os	Histogram based on I/O times giving total count and individual read/write counts.
Per disk I/Os	Histogram based on I/O times giving total count and individual read/write counts.
Max concurrent I/Os	Maximum I/Os running concurrently.
Disk utilization	Per disk utilization over the duration of the trace showing I/Os handled.
Number of sectors	Histogram showing the number of sectors (in 512-byte increments) written/read.
I/O throughput	Histogram showing M/sec of reads/writes from first I/O to last.
RPC responsiveness	Histogram of any RPC that was called during the trace.

fsstruct.awk for AIX and fsstructlx.awk for Linux

The purpose of these scripts is to summarize events either from **`errpt -a`** from AIX or `/var/log/messages` from Linux. The scripts can either gather mmfs-specific or general events and create a one-line time-stamped entry for each.

The output generated by these **`.awk`** scripts can help identify where an mmfs problem started. For instance, an unmount of a file system may have started as an NSD failure on nodeA and propagated to the other nodes as an error `E_NO_MGR (rc=212)`. Being able to identify which node was first to report a problem can help point to what the real issue is.

Because of where mmfs errors are reported in various locations, if the cluster contains mixed operating systems, the use of working collectives are needed when collecting data to take this into account.

Although these scripts can be run on a single node, they are meant to take these error messages' log files from multiple or all nodes in the cluster and condense all the entries to allow sorting.

For example, the following commands collect error log information coming from multiple nodes:

- On AIX:

  ```
  mmdsh -N [all|aix.wcoll] "errpt -a" > errpt.out
  ```

- On Linux:

  ```
  mmdsh -N [all | linux.wcoll] 'cat /var/log/messages' > messages.out
  ```

To create an mmfs-only summary, use the following command:

```
/usr/lpp/mmfs/samples/debugtools/fsstruct[lx].awk errpt.out | sort -k 1,1 >
all.gpfs
```

To get a summary from both the operating system and Spectrum Scale from the same file, use the following command:

```
/usr/lpp/mmfs/samples/debugtools/fsstruct[lx].awk all=1 errpt.out | sort -k 1,1 >
all.err
```

8.9 Deadlock detection features

Starting with Spectrum Scale 4.1, a new feature to automate deadlock detection, deadlock data collection, and deadlock breakup was introduced to help the administrators when facing a Spectrum Scale deadlock situation. It is referred as *deadlock amelioration*.

A deadlock situation can occur when one or more nodes are waiting for some information and this information is not available caused mostly due to a network or disk failure or even for some application problem.

The complex nature of the Spectrum Scale locking infrastructure due to its parallelism and concurrent access, the dependency on the proper operation of disks and networks, and the overall complexity of operating in a clustered environment can contribute to increase the probability of a deadlock condition.

Deadlocks can be disruptive in certain situations and a deadlock can represent a single point of failure that can lead to a cluster failure. When a deadlock is encountered on a production system, it can take a long time to debug. The typical approach to recovering from a deadlock involves rebooting all of the nodes in the cluster. Thus, deadlocks can lead to prolonged and complete outages of clusters. Some of the typical symptoms of a deadlock condition are:

- Application processes hung and usually cannot be killed, even with SIGKILL (kill -9).
- The **`ls -l`** or **`df`** hangs and cannot be killed.
- mmfsd threads with long wait times in mmfsadm dump waiters.
- Nodes either completely or partially unresponsive (console sessions hung, remote commands cannot be executed).
- The response times start to slow down, even though some progress is being made, the system looks effectively deadlocked to users.

To troubleshoot deadlocks, some specific types of debug data must be collected while the deadlock is in progress. Data collection commands must be run manually, and if this is not done before the deadlock is broken up, determining the root cause of the deadlock after that will be difficult.

Following are the new enhancements or the deadlock detection process introduced on Spectrum Scale 4.1:

- 8.9.1, “Automated deadlock detection” on page 450
- 8.9.2, “Automated deadlock data collection” on page 451
- 8.9.3, “Automated deadlock breakup” on page 452

8.9.1 Automated deadlock detection

Many deadlocks involve long waiters; for example, mmfsd threads that have been waiting for some event for a considerable duration of time. With some exceptions, long waiters typically indicate that something in the system is not healthy. There may be a deadlock in progress, some disk might be failing, or the entire system might be overloaded.

All waiters can be broadly divided into four categories:

- Waiters that can occur under normal operating conditions and can be ignored by automated deadlock detection.
- Waiters that correspond to complex operations and can legitimately grow to moderate lengths.

- Waiters that should never be long. For example, most mutexes should only be held briefly.
- Waiters that can be used as an indicator of cluster overload. For example, waiters waiting for I/O completions or network availability.

Automated deadlock detection monitors waiters. Deadlock detection relies on a configurable threshold to determine if a deadlock is in progress. When a deadlock is detected, an alert is issued in the *mmfs.log*, the operating system log, and the *deadlockDetected* callback is triggered.

Automated deadlock detection is enabled by default and controlled with the **mmchconfig** attribute `deadlockDetectionThreshold`. A potential deadlock is detected when a waiter waits longer than `deadlockDetectionThreshold`. To view the current threshold for deadlock detection, enter the command as shown in Example 8-35.

Example 8-35 The mmlsconfig with deadlockDetectionThreshold parameter

```
root@fsaix1:/>mmlsconfig deadlockDetectionThreshold
deadlockDetectionThreshold 300
```

To disable automated deadlock detection, specify a value of 0 for the `deadlockDetectionThreshold` attribute as shown in the Example 8-36.

Example 8-36 Using mmchconfig to disable automatic deadlock detection

```
root@fsaix1:/>mmchconfig deadlockDetectionThreshold=0
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

8.9.2 Automated deadlock data collection

Usually when troubleshooting a typical deadlock problem, the following data must be collected:

- A full internal dump (**mmfsadm dump all**)
- A dump of kthreads (**mmfsadm dump kthreads**)
- Trace snapshot (10 - 30 seconds of trace data)

The Spectrum Scale 4.1 added the automated deadlock data collection feature so it can be used to help during the process of gathering vital debug data for detecting potential deadlock problems.

Automated deadlock data collection is enabled by default and controlled with the **mmchconfig** attribute `deadlockDataCollectionDailyLimit`. This attribute specifies the maximum number of times debug data can be collected in a 24-hour period. To view the current data collection interval, enter the command as shown in Example 8-37.

Example 8-37 Using mmlsconfig to display deadlock data collection values

```
root@fsaix1:/>mmlsconfig deadlockDataCollectionDailyLimit
deadlockDataCollectionDailyLimit 10
```

To change this attribute the **mmchconfig deadlockDataCollectionDailyLimit=value** command can be used to any value 1 - 2147483647.

To disable automated deadlock data collection, specify a value of 0 for the `deadlockDataCollectionDailyLimit`.

8.9.3 Automated deadlock breakup

Automated deadlock breakup can help the administrators to resolve a deadlock situation without human intervention. To break up a deadlock, less disruptive actions are tried first; for example, causing a file system panic. If necessary, more disruptive actions are then taken; for example, shutting down a Spectrum Scale mmfsd daemon. If a system administrator prefers to control the deadlock breakup process, the `deadlockDetected` callback can be used to notify system administrators that a potential deadlock has been detected.

The information from the **`mmdiag --deadlock`** section can then be used to help determine what steps to take to resolve the deadlock when this situation has been reached.

The Automated deadlock breakup is disabled by default and controlled with **`mmchconfig attribute deadlockBreakupDela`**y.

The **`deadlockBreakupDelay`** attribute specifies how long to wait after a deadlock is detected before attempting to break up the deadlock. Enough time must be provided to allow the debug data collection to complete. To view the current breakup delay, enter the following command:

```
mmlsconfig deadlockBreakupDelay
```

The system displays output similar to the following:

```
deadlockBreakupDelay 0
```

The value of 0 shows that automated deadlock breakup is disabled. To enable automated deadlock breakup, specify a positive value for `deadlockBreakupDelay`. If automated deadlock breakup is to be enabled, a delay of 300 seconds or longer is suggested.

8.10 Additional information

When trying to analyze any Spectrum Scale problem, depending on the nature of the failure, the system can become unresponsible and even the data collection can be affected.

Following are some of the problems that can affect the system administrators during the problem determination.

Spectrum Scale file system hangs

The general data collection steps when a Spectrum Scale file system hang occurs are as follows:

1. Gather waiters from all nodes (see 8.7.1, “Collecting specific information” on page 439).
2. From all node with waiters, use the **`mmfsadm dump all`** command.
3. If any remote clusters exist, perform steps 1 and 2 from those clusters.
4. Use **`gpfs.snap`** from one node; use **`gpfs.snap -z`** from all other nodes.

Spectrum Scale-related command hangs

The general data collection steps when file-system-related commands or scripts hang (such as, **`ls`**, **`cp`**, **`mv`**, **`mkdir`**, and so on) are as follows:

1. Check for long waiters. If waiters exist, treat the problem as a possible hang.
2. If no waiters exist, continue and re-create the capturing of the trace.

3. Ensure that the trace file size and trace buffer size are enough.
4. Start the mmfs tracing with the **mmtrace** command.
5. Re-create the error.
6. Stop mmfs tracing.
7. Gather the data with the **mmfsadm dump all** command.

Spectrum Scale administration command hangs

The general data error collection steps when a Spectrum Scale administration command hangs are as follows:

1. Run "*script* /tmp/mmtrace.out" (script is a command that collects every window output)
2. Re-create the capturing of the trace
3. Set the export DEBUG=1
4. Re-create the error again
5. Exit the trace
6. Run the **gpfs.snap** script
7. Unexport the DEBUG=1 variable with "*unset DEBUG'"*
8. Stop the "*script"* command with the **exit** command or issuing a kill against the process ID number

8.11 IBM Spectrum Scale problem determination scenarios

Given the data collection and tools described in previous sections, this section provides some examples of common Spectrum Scale-related problems and how to collect and analyze the gathered data. Not all problems are covered in this chapter.

8.11.1 Scenario 1: Spectrum Scale daemon not running

There are several indications to identify that the Spectrum Scale daemon might not be running. In this section, we cover some of the most common indications from a down Spectrum Scale daemon and some steps to follow in order to correct the problem.

The indications can be any of the following:

- The file system has been enabled to mount automatically, but the mount has not completed.
- You issue a Spectrum Scale command and receive the message:

  ```
  6027-665 Failed to connect to file system daemon: Connection refused.
  ```

- The Spectrum Scale log *does not* contain the message:

  ```
  6027-300 [N] mmfsd ready
  ```

- The Spectrum Scale log file contains this error message:

  ```
  'Error: daemon and kernel extension do not match.'
  ```

  This error indicates that the kernel extension currently loaded in memory and the daemon currently starting have mismatching versions. This situation may arise if a Spectrum Scale code update has been applied, and the node has not been rebooted before starting Spectrum Scale. To recover from this error, ensure that all Spectrum Scale file systems are successfully unmounted, and reboot the node. The **mmlsmount** command can be used to ensure that all file systems are unmounted. In some situations, the **mmbuildgpl** command (as described in 3.4.3, "Building the IBM Spectrum Scale portability layer on Linux nodes" on page 102) can be executed to recompile Spectrum Scale kernel extensions if some upgrade was executed, followed by a node reboot.

Steps to follow when the Spectrum Scale daemon is not active

If you encounter any of the preceding situations, we suggest you to verify if the Spectrum Scale daemon is running by executing the following procedures:

1. Make sure that the Spectrum Scale daemon is active by issuing:

   ```
   ps -e | grep mmfsd
   ```

   If the daemon is not active, the **ps -ef** command will not show anything, but if the daemon is active, the output should be similar to the following:

   ```
   root 18612430 11010232   0   Oct 13      -  1:15 /usr/lpp/mmfs/bin/mmfsd
   ```

   If the output does not show this, the Spectrum Scale daemon needs to be started with the **mmstartup** command.

2. When the autoload option is not specified on the **mmcrcluster** or the **mmchconfig** command, you need to manually start the daemon by issuing the **mmstartup** command. When using autoload for the first time, **mmstartup** must be run manually. The autoload takes effect on the next reboot.

3. Verify that the Spectrum Scale cluster network is operational by issuing the ping command:

   ```
   ping nodename
   ```

   Make sure that every Spectrum Scale node is able to reply ping requests without loosing packets.

4. Verify that the Spectrum Scale cluster configuration data is available by looking in the Spectrum Scale log. If the message `6027-1592 Unable to retrieve GPFS cluster files from node nodeName` is displayed, make sure that the mentioned node is accessible.

5. Verify that the Spectrum Scale environment is properly initialized by issuing these commands and ensuring that the output is as expected:

   – Issue the **mmlscluster** command to list the cluster configuration. This will also update the Spectrum Scale configuration data on the node. Correct any reported errors before continuing.

   – List all file systems that were created in this cluster. For an AIX node, issue:

     ```
     lsfs -v mmfs
     ```

     For a Linux node, issue:

     ```
     cat /etc/fstab | grep gpfs
     ```

   If any of these commands produce unexpected results, this may be an indication of corrupted Spectrum Scale cluster configuration data. Contact the IBM Support Center.

6. Spectrum Scale requires a quorum of nodes to be active before any file system operations can be performed. This requirement guarantees that a valid single token management domain exists for each Spectrum Scale file system. Before the existence of a quorum, most requests are rejected with a message indicating that quorum does not exist. To identify which nodes in the cluster have daemons up or down, issue:

   ```
   mmgetstate -aL
   ```

   If insufficient nodes are active to achieve quorum, go to any nodes not listed as active and perform problem determination steps on these nodes. A quorum node indicates that it is part of a quorum by writing an mmfsd ready message to the Spectrum Scale log. Remember that your system may have quorum nodes and non-quorum nodes, and only quorum nodes are counted to achieve the quorum.

7. This step applies only to AIX nodes.

Verify that Spectrum Scale kernel extension is not having problems with its shared segment by invoking:

```
cat /var/adm/ras/mmfs.log.latest
```

Messages such as "`6027-319 Could not create shared segment`" must be corrected by the following procedure:

a. Issue the **mmshutdown** command

b. Issue the **mmfsadm** cleanup command

c. If you are still unable to resolve the problem, reboot the node.

8. This step applies only to Linux nodes.

   When installing Spectrum Scale on a Linux machine, you must build the Spectrum Scale portability layer binary files based on the kernel configuration of your system. Some of the problems that can occur due to kernel extensions or compatibility layer installations are:

   a. If the portability layer is not built, you may see messages similar to:

   ```
   Wed Oct 16 15:56:30 EDT 2014: runmmfs starting
   Removing old /var/adm/ras/mmfs.log.* files:
   Unloading modules from /lib/modules/2.6.32.12-0.6-ppc64/extra
   runmmfs: The /lib/modules/2.6.32.12-0.6-ppc64/extra/mmfslinux.ko kernel
   extension does not exist.
   runmmfs: Unable to verify kernel/module configuration.
   Loading modules from /lib/modules/2.6.32.12-0.6-ppc64/extra
   runmmfs: The /lib/modules/2.6.32.12-0.6-ppc64/extra/mmfslinux.ko kernel
   extension does not exist.
   runmmfs: Unable to verify kernel/module configuration.
   Wed Oct 16 15:56:30 EDT 2014 runmmfs: error in loading or unloading the mmfs
   kernel extension
   Wed Oct 16 15:56:30 EDT 2014 runmmfs: stopping GPFS
   ```

   b. The Spectrum Scale kernel modules, `mmfslinux` and `tracedev`, are built with a kernel version that differs from that of the currently running Linux kernel. This situation can occur if the modules are built on another node with a different kernel version and copied to this node, or if the node is rebooted using a kernel with a different version.

   c. If the `mmfslinux` module is incompatible with your system, you may experience a kernel panic on Spectrum Scale start. Ensure that the Spectrum Scale has been built and installed properly.

9. If the previous Spectrum Scale daemon was brought down and you are trying to start a new daemon but are unable to, this is an indication that the original daemon did not completely go away. Go to that node and check the state of Spectrum Scale. Stopping and restarting Spectrum Scale or rebooting this node will often return Spectrum Scale to normal operation.

8.11.2 Scenario 2: Analyzing waiters

This section gives a basic technique for analyzing waiters that can help the problem determination process in finding the cause or causes of a file system hang or performance issue that is occurring or has occurred on the Spectrum Scale cluster. The intention is to provide a high-level overview at what the significant waiters are and what information can be gleaned from them.

The presumption is that waiters have been gathered from all nodes, including nodes in the local cluster and any remote clusters that remotely mount file systems.

Note: Long running waiters are usually indicative of a slow or failed disk, a slow network, or a slow CPU.

Removing secondary waiters

This process is a general practice to first remove waiters that are considered *secondary* to any problem that is occurring.

The following list provides information from the Spectrum Scale waiters that might help with identifying the waiters that can be ignored and removed from the analysis. Typically, the text follows the *reason* portion of a waiter. Example 8-38 shows where the reason portion is found in a typical Spectrum Scale waiter.

Example 8-38 Showing the waiter information

```
0x40001D5B6C0 waiting 5628.211637314 seconds, Create handler: on ThCond
0x18002F07AD0 (0xD000000002F07AD0) (LkObj), reason 'change_lock_shark waiting to
set acquirePending flag
```

- Make a copy of the original waiters file to use for this process.

  Remove waiters from the waiters file that have the following reasons:

  - `change_lock_shark waiting to set acquirePending flag`
  - `waiting for <lock-type> lock where <lock-type> can be LX, RF, RS, RO, XW, WW`
  - `waiting because of local byte range lock conflict`
  - `waiting for fetch-n-lock to complete`
  - `RPC wait for tmMsgTellAcquire1 on node xxx.xxx.xxx`
  - `waiting because of local byte range lock conflict`
  - `wait for SubToken to become stable`
  - `waiting until pit work is complete`
  - `waiting to set acquirePending flag`
  - `waiting for helper threads`
  - `tmMsgTellAcquire`
  - `tmMsgRevoke`
  - `waiting until pit work is complete`
  - `waiting to reap fsck pointer`
  - `waiting to find big enough file for helper`
  - `waiting for permission to run`
  - `waiting for PIT_Stop or restart message from master`
  - `waiting for work`

Creating waiters hierarchy

After the secondary waiters are removed, a hierarchy can be applied to the remaining waiters. This hierarchy is a list of waiter types that is traversed from top to bottom, looking for a match. After a match is found, these waiter types can point to areas such as node recovery, I/O, networking, or tuning to investigate. They can also reveal a Spectrum Scale deadlock or simply a heavily used Spectrum Scale file system.

- Based on the following ordered list, organize the remaining waiters into the categories listed, looking for matches to the given text of each waiters text

  a. Node Recovery

  Look for the following waiters:

  - GroupProtocolDriverThread
  - RPC wait for ccMsgGroupLeave
  - RPC wait for sgmMsgExeTMPhase

- MMFS group recovery phase
- waiting for stripe group takeover

These types of waiters point to recovery being done in the cluster. If a node or nodes have left or are joining the cluster, RPCs exist that all nodes must respond to. If one (or more) nodes do not respond, activity on the cluster will hang until responses are received.

RPCs are interprocess communications that allow mmfs daemons to transfer data to and from each other by using multiple threads and one or more processes.

Perform the following action:

i. Node recovery is driven by the Cluster Manager node. Use the **mmlsmgr** command to locate the Cluster Manager.

ii. Use the **mmdiag --network** (GPFS-3.4) or **mmfsadm dump tscomm** command on the Cluster Manager to locate the nodes that have "Pending messages."

iii. Check these nodes for network and resource or performance issues.

iv. If a problem is to be opened with IBM Service, gather the default documentation that is outlined in "Scenario 1: Spectrum Scale daemon not running" on page 453.

b. Local Disk I/O

Look for the following waiters:

- NSD I/O worker: for I/O completion on disk
- Mount handler: for open disk device

These waiters appear on NSD server nodes and can indicate an issue in the disk subsystem depending on their length.

Perform the following action:

i. Use system tools such as **iostat** or **topas** to monitor the output for any abnormal disk activity for the disks and review the system error logs for AIX, Linux, or Windows operating systems along with the event logs for disk subsystem and SAN for possible causes.

ii. If a problem is to be opened with IBM Service, gather the default documentation that is outlined in "Scenario 1: Spectrum Scale daemon not running" on page 453.

c. Network I/O

Look for the following waiters:

- RPC wait for NSD I/O completion on
- RPC wait for getData on node <xxx.xxx.xxx>
- waiting for exclusive use of connection

These waiters point to an issue in the network between the client node and the NSD server that they point to. These waiters may be a secondary issue if there are also long waiters waiting on Local Disk I/O. Investigate Local Disk I/O waiters first.

Perform the following action:

i. Check the network with commands such as **netstat -D** or **netstat -s**, looking for packet drops or retransmit errors in the TCP section. All network switches are between the client and the NSD server node.

ii. Look at the system error logs for AIX, Linux, or Windows operating system for network-related errors.

iii. If a problem is to be opened with IBM Service, gather the default data that is outlined in 8.11.1, "Scenario 1: Spectrum Scale daemon not running" on page 453.

d. Token Revoke

   Look for the following waiter:

   - RPC wait for tmMsgRevoke on node <xxx.xxx.xxx>

   This waiter happens when one node is waiting on a resource that is held by another. There may be multiple nodes that have waiters waiting on *tmMsgRevokes*. Many times one node can be identified as the source of the contention as the *tmMsgRevoke* point to one node or one node points to the next until the source is found.

   Perform the following action:

   i. Gather the default documentation that is outlined in 8.11.1, “Scenario 1: Spectrum Scale daemon not running” on page 453.

   ii. Determine the node that is holding the resources deadlock by examining the *tmMsgRevoke message(s)* and the IP address they point to. If one node can clearly be identified as the source, issue the **`mmfsadm cleanup`** command on that node. This step causes the mmfs daemon to recycle.

e. Other waiters

   Look for the following waiters:

   - stealEngine loop wrapping
   - In kernel waiting to quiesce

   These waiters do not have the same cause in common and is why they are placed in the “Other waiters” category.

   Perform the following action:

   i. The “stealEngine loop wrapping” waiter usually is present if the page pool is to set small and there are not enough buffers in the page pool to work with. Increase the page pool with the **`mmchconfig`** command (using the **-l** or **-i** option can alleviate this issue).

      The “In kernel waiting to quiesce” waiter usually results from the **`mmcrsnapshot`**, **`mmdelsnapshot`**, **`mmdelfileset`**, or **`mmfsctl suspend`** commands.

   ii. If a problem is to be opened with IBM Service, gather the default documentation that is outlined in 8.11.1, “Scenario 1: Spectrum Scale daemon not running” on page 453.

f. Long waiters

   Action: Gather the documentation that is listed in the “Gathering the waiters” on page 443.

► Start the investigation with the category that is highest on the ordered list.

See 8.11.3, “Scenario 3: Spectrum Scale file system hang” on page 458 for an example of applying the rules given a hung file system.

8.11.3 Scenario 3: Spectrum Scale file system hang

The goal of this scenario is to determine the node or nodes that are causing the hang situation and gather all the documentation possible.

Example 1

A node in a local cluster remotely mounts its file system from a remote cluster. The external symptom is a file system hang. Perform the following steps:

1. Using the procedures described in “Scenario 2: Analyzing waiters” on page 455, view the waiters from all the nodes. See Example 8-39.

Example 8-39 Viewing the waiters

```
usa: 0x108395A0 waiting 491.816423025 seconds, SharedHashTab fetch handler: on ThCond
0x1077FEB0 (0x1077FEB0) (MsgRecord), reason 'RPC wait' for tmMsgTellAcquire1 on node
192.168.101.142
usa: 0x10819CA0 waiting 6751.419359689 seconds, Writebehind worker: on ThCond 0x400025491B0
(0x400025491B0) (MsgRecord), reason 'RPC wait' for NSD I/O completion
usa: 0x40000918260 waiting 6752.659368250 seconds, Writebehind worker: on ThCond 0x1086C5A0
(0x1086C5A0) (MsgRecord), reason 'RPC wait' for NSD I/O completion
usa: 0x40000916FB0 waiting 6750.199351265 seconds, Writebehind worker: on ThCond usa:
0x107B6950 waiting 6750.309352025 seconds, Writebehind worker: on ThCond 0x400010ADB60
(0x400010ADB60) (MsgRecord), reason 'RPC wait' for NSD I/O completion
usa: 0x1065A450 waiting 6748.259337872 seconds, Sync handler: on ThCond 0x180030BE068
(0xD0000000030BE068) (LkObj), reason 'waiting for WW lock'
germany:0x11181C630 waiting 6008.507415856 seconds, NSD I/O Worker: on ThCond 0x11709EB60
(0x11709EB60) (MsgRecord), reason 'RPC wait' for getData onnode 192.168.101.132
```

2. Using the suggestions described in “Removing secondary waiters” on page 456, first remove the *secondary* waiters from all the waiters that are gathered as shown in Example 8-40.

Example 8-40 Removing the secondary waiters

```
usa: 0x108395A0 waiting 491.816423025 seconds, SharedHashTab fetch handler: on ThCond
0x1077FEB0 (0x1077FEB0) (MsgRecord), reason 'RPC wait' for tmMsgTellAcquire1 on node
192.168.101.142
usa: 0x1065A450 waiting 6748.259337872 seconds, Sync handler: on ThCond 0x180030BE068
(0xD0000000030BE068) (LkObj), reason 'waiting for WW lock'
```

3. Using the hierarchy from the list, look for each type of waiter to help classify where the problem might be. The `'RPC wait' for NSD I/O completion` waiter on usa server node says that an I/O is waiting in the network for its read or write processing to complete. The `'RPC wait' for getData` waiter on the germany server node says that it is waiting to get data from node 192.168.101.132, which is the usa node. These nodes are waiting for each other and is the result cause of the hang condition. See Example 8-41.

Example 8-41 Finding the reason of the hang condition by using the waiter

```
usa: 0x10819CA0 waiting 6751.419359689 seconds, Writebehind worker: on ThCond 0x400025491B0
(0x400025491B0) (MsgRecord), reason 'RPC wait' for NSD I/O completion
usa: 0x40000918260 waiting 6752.659368250 seconds, Writebehind worker: on ThCond 0x1086C5A0
(0x1086C5A0) (MsgRecord), reason 'RPC wait' for NSD I/O completion
usa: 0x40000916FB0 waiting 6750.199351265 seconds, Writebehind worker: on ThCond usa:
0x107B6950 waiting 6750.309352025 seconds, Writebehind worker: on ThCond 0x400010ADB60
(0x400010ADB60) (MsgRecord), reason 'RPC wait' for NSD I/O completion
germany:0x11181C630 waiting 6008.507415856 seconds, NSD I/O Worker: on ThCond 0x11709EB60
(0x11709EB60) (MsgRecord), reason 'RPC wait' for getData onnode 192.168.101.132
```

4. In hang conditions where RPC waits are occurring, the starting point for further investigation is the dump tscomm stanza from the **`dump all`** command that has been gathered from each of the nodes that have waiters.

Start with usa node. The dump tscomm stanza from usa (part of it is in Example 8-42) shows that the dest of the ***pending*** **nsdMsgWrite** is **192.168.101.142**, which correlates to the germany node in the remote cluster.

Example 8-42 The dump tscomm stanza output

```
===== dump tscomm =====
Pending messages:
  msg_id 581893, service 16.1, msg_type  1 'nsdMsgWrite', n_dest 1, n_pending 1
  this 0x1081F2F0, n_xhold 1, ccP 0x1800276BBB0 cbFn 0x0
    sent by 'Writebehind worker' (0x1086B180)
    dest 192.168.101.142 status pending   , err 0, reply len 0
  msg_id 581895, service 16.1, msg_type  1 'nsdMsgWrite', n_dest 1, n_pending 1
  this 0x400010B6BB0, n_xhold 1, ccP 0x1800276BBB0 cbFn 0x0
    sent by 'Writebehind worker' (0x10842890)
    dest 192.168.101.142 status pending   , err 0, reply len 0
  msg_id 581385, service 16.1, msg_type  1 'nsdMsgWrite', n_dest 1, n_pending 1
  this 0x400025482A0, n_xhold 1, ccP 0x1800276BBB0 cbFn 0x0
    sent by 'Writebehind worker' (0x10824210)
    dest 192.168.101.142 status pending   , err 0, reply len 0
  msg_id 581386, service 16.1, msg_type  1 'nsdMsgWrite', n_dest 1, n_pending 1
  this 0x400010AD3D0, n_xhold 1, ccP 0x1800276BBB0 cbFn 0x0
    sent by 'Writebehind worker' (0x10875B90)
    dest 192.168.101.142 status pending   , err 0, reply len 0
```

Because all pending messages point to **192.168.101.142 (germany)**, the investigation now moves to the node germany in the remote cluster. The dump tscomm from germany is shown in Example 8-43.

Example 8-43 Pending messages from dump tscomm

```
===== dump tscomm =====

Pending messages:
  msg_id 16372342, service 13.1, msg_type  1 'tmMsgRevoke', n_dest 1, n_pending 1
  this 0x1170CA750, n_xhold 0, ccP 0xF1000003E0D97A18 cbFn 0x11007A998
    sent by 'tmServerSideRevokeThread' (0x110299310)
    dest 192.168.101.132 status pending   , err 0, reply len 0
  msg_id 16372343, service 13.1, msg_type  1 'tmMsgRevoke', n_dest 1, n_pending 1
  this 0x1170C2C970, n_xhold 0, ccP 0xF1000003E0D97A18 cbFn 0x11007A998
    sent by 'tmServerSideRevokeThread' (0x110299310)
    dest 192.168.101.132 status pending   , err 0, reply len 0
  msg_id 16335429, service 0.0, msg_type 252 'getData', n_dest 1, n_pending 1
  this 0x1170E9F0, n_xhold 1, ccP 0xF1000003E0D97A18 cbFn 0x0
    sent by 'NSD I/O Worker' (0x11181C630) dest 192.168.101.132 status pending   ,
err 0, reply len 2097152
```

The example shows that all pending messages are pointing to **192.168.101.132 (usa)**, which confirms that the issue appears to be in the network but does not identify root cause.

5. Check the error logs (**errpt -a** or /var/log/messages), **netstat** outputs, and switch logs for possible causes. Re-creating and running **iptrace** or **tcpdump**, as needed, might be necessary.

   This hang condition was cleared by recycling the mmfs daemon on one of the nodes. Because it was only a client node, **mmshutdown** and **mmstartup** was performed on node usa and the hang was resolved.

Example 2

Further evidence that this issue was network-related can be found by first identifying the socket to which 192.168.101.132 is attached. This information is in the tscomm stanza, in the `Connections between TS nodes` portion (Example 8-44). In this case, it is socket 14.

Example 8-44 Connection table from dump tscomm

```
Connections between nodes:
        node destination     status     err b  sock      sent     recvd   sseq   rseq  retry   ver  ostype
      <c1n1> 192.168.101.132 connected    0 -    14     27362     27363  27350  27363      0  1202  AIX  /B
```

By using the dump tscomm stanza again (Example 8-45), look in the `Message receiver state` portion for the `Receiver` threads that reference the socket identified previously (socket 14) and the `msg_id`. This information reveals that this socket had not received all the bytes of data it was expecting.

Example 8-45 Message receiver state from dump tscomm

```
Message receiver state:
Receiver 553325
    sock 14: reading data, expecting 2097152 bytes, received 1924393, type reply, msg_id 16335429
```

8.11.4 Scenario 4: Spectrum Scale file system unmounting

The starting point for analyzing unmounts is the mmfs logs on the node where the unmount occurred. This node however, might not be the original source where the initial unmount was manifested. The mmfs logs from all the nodes must be gathered and investigated to determine which node initially unmounted and what caused the initial error.

Example 1

While attempting to use the **dd** command to write to the file system on node slovenia, a `Stale NFS file handle message` is received before completion of the command. See Example 8-46.

Example 8-46 Initial file system error

```
# dd if=/dev/zero of=/testmigr/test.ouptut bs=256K count=10000
dd: writing `/testmigr/test.ouptut': Stale NFS file handle
817+0 records in
816+0 records out
213909504 bytes (214 MB) copied, 2.99825 seconds, 71.3 MB/s
```

Perform the following steps:

1. Look first at the mmfs logs on slovenia. The cause for the **unmount** is that too many disks are unavailable. See Example 8-47.

Example 8-47 Portion of mmfs.log.latest from slovenia

```
Tue Jul 13 09:56:53.389 2010: File System testmigr unmounted by the system with
return code 217 reason code 0
Tue Jul 13 09:56:53.390 2010: Too many disks are unavailable.
Tue Jul 13 09:56:53.391 2010: File system manager takeover failed.
Tue Jul 13 09:56:53.390 2010: Too many disks are unavailable.
Tue Jul 13 09:56:53 EDT 2010: mmcommon preunmount invoked.  File system: testmigr
Reason: SGPanic
```

2. Because the problem that is revealed in the mmfs logs says that too many disks are unavailable, determine the status of the NSDs in the file system, with **mmlsdisk**. Example 8-48 shows that two disks are down.

Example 8-48 mmlsdisk out from /testmigr file system

```
# mmlsdisk testmigr
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
nsdhdisk2    nsd         512      60 yes      yes   ready         up           system
nsdhdisk3    nsd         512      60 yes      yes   ready         down         system
nsdhdisk4    nsd         512      80 yes      yes   ready         down         system
nsdhdisk5    nsd         512      80 yes      yes   ready         up           system
```

Evidence that further investigation is needed is that the node slovenia has no direct or local access to the NSDs in the `/dev/testmigr` file system. See Example 8-49.

Example 8-49 mmlsnd output showing slovenia as an NSD client

```
# mmlsnsd -m -f testmigr

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 nsdhdisk2    C0A865844C34E4FC   /dev/hdisk1    germany                  server node
 nsdhdisk2    C0A865844C34E4FC   /dev/hdisk7    usa                      server node
 nsdhdisk3    C0A865844C34E4FE   /dev/hdisk2    germany                  server node
 nsdhdisk3    C0A865844C34E4FE   /dev/hdisk8    usa                      server node
 nsdhdisk4    C0A865844C34CCD0   /dev/hdisk3    germany                  server node
 nsdhdisk4    C0A865844C34CCD0   /dev/hdisk9    usa                      server node
 nsdhdisk5    C0A865844C34CCD2   /dev/hdisk4    germany                  server node
 nsdhdisk5    C0A865844C34CCD2   /dev/hdisk10   usa                      server node
```

3. Gather mmfs logs from the node. See 8.6.2, “IBM Spectrum Scale severity tags” on page 429.

Collecting logs from all nodes in the cluster reveals that **unmount** started on node germany and propagated to all nodes in the cluster because of a disk problem on node germany. See Example 8-50.

Example 8-50 mmfs logs from time of unmount

```
germany:  Tue Jul 13 09:56:00.819 2010: File System testmigr unmounted by the system with
return code 217 reason code 0
germany:  Tue Jul 13 09:56:00.828 2010: Too many disks are unavailable.
germany:  Tue Jul 13 09:56:00.836 2010: File system manager takeover failed.
germany:  Tue Jul 13 09:56:00.852 2010: Too many disks are unavailable.
germany:  Tue Jul 13 09:56:01 EDT 2010: mmcommon preunmount invoked.  File system: testmigr
Reason: SGPanic
usa:  Tue Jul 13 09:56:51 EDT 2010: mmcommon preunmount invoked.  File system: testmigr
Reason: SGPanic
usa:  Tue Jul 13 09:56:51.437 2010: Recovery Log I/O Failed, Unmounting file system
testmigr
usa:  Tue Jul 13 09:56:51.451 2010: Too many disks are unavailable.
usa:  Tue Jul 13 09:56:51.459 2010: File System testmigr unmounted by the system with
return code 218 reason code 0
usa:  Tue Jul 13 09:56:51.467 2010: Too many disks are unavailable.
usa:  Tue Jul 13 09:56:51.489 2010: Node 192.168.101.132 (usa) resigned as manager for
testmigr.
usa:  Tue Jul 13 09:56:51.507 2010: Too many disks are unavailable.
```

```
usa:  Tue Jul 13 09:56:51.516 2010: Node 192.168.101.142 (germany) appointed as manager for
testmigr.
usa:  Tue Jul 13 09:56:51.582 2010: Node 192.168.101.142 (germany) resigned as manager for
testmigr.
usa:  Tue Jul 13 09:56:51.599 2010: Too many disks are unavailable.
usa:  Tue Jul 13 09:56:51.609 2010: Node 192.168.101.133 (slovenia) appointed as manager
for testmigr.
slovenia:  Tue Jul 13 09:56:53 EDT 2010: mmcommon preunmount invoked.  File system:
testmigr  Reason: SGPanic
slovenia:  Tue Jul 13 09:56:53.389 2010: File System testmigr unmounted by the system with
return code 217 reason code 0
slovenia:  Tue Jul 13 09:56:53.390 2010: Too many disks are unavailable.
slovenia:  Tue Jul 13 09:56:53.390 2010: Too many disks are unavailable.
slovenia:  Tue Jul 13 09:56:53.391 2010: File system manager takeover failed.
```

4. Investigation of what the disk issue might be points to node germany. Review system logs (**errpt -a**) and any relevant SAN or disk subsystem logs for errors. In this case, **errpt -a** showed that node germany received SC_DISK_SDDAPPCM_ER and SC_DISK_ERR7 on nsdhdisk2, and nsdhdisk4.

Example 2

In this example, the unmount occurs during heavy system activity while running a backup of the file system. The backup fails with `insufficient memory (ENOMEM)` messages and the file system unmounts.

Perform the following steps:

1. Look at the mmfs logs on usa at roughly the reported time of the failure. Example 8-51 shows Signal 11 (SIGSEGV) and then Signal 6 (SIGABORT). Signal 11 is a memory violation, which indicates that Spectrum Scale tried to get or use memory and was denied. Signal 6 is the mmfsd, generating the assert.

Example 8-51 The mmfs logs from usa showing mmfs daemon assert

```
Mon Jul 12 14:29:46.912 2010: Signal 11 at location 0x40050B10 in process 884, link reg
0xFFFFFFFFFFFFFFFF.
Mon Jul 12 14:29:46.913 2010: rax    0x0000000000000000  rbx    0x0000000000000038
Mon Jul 12 14:29:46.912 2010: rcx    0x0000001010FA35D8  rdx    0x0000000000000000
Mon Jul 12 14:29:46.913 2010: rsp    0x00007FFF12A7DE58  rbp    0x0000000000000000
Mon Jul 12 14:29:46.912 2010: rsi    0x0000000000000089  rdi    0x0000000000000000
Mon Jul 12 14:29:46.913 2010: r8     0x00007FFF12A7DE14  r9     0x0000001010FA3618
Mon Jul 12 14:29:46.912 2010: r10    0x0000001010FA3618  r11    0x0000000000000001
Mon Jul 12 14:29:46.913 2010: r12    0x0000000000000089  r13    0x000000101120DD34
Mon Jul 12 14:29:46.912 2010: r14    0x00000010112D6A08  r15    0x000000000000000C
Mon Jul 12 14:29:46.913 2010: rip    0x0000000040050B10  eflags 0x0000000000010202
Mon Jul 12 14:29:46.912 2010: csgsfs 0x0000000000000033  err    0x0000000000000006
Mon Jul 12 14:29:46.913 2010: trapno 0x000000000000000E
Mon Jul 12 14:29:47.511 2010: Traceback:
Mon Jul 12 14:29:47.513 2010: 0:0000000040050B10 BaseCacheObj::BaseCacheObj().40050AF0 + 20
Mon Jul 12 14:29:47.514 2010: 1:0000000040050BCD CacheObj::CacheObj(MutexName).40050BB0 + 1D
Mon Jul 12 14:29:47.513 2010: 2:00000000400A2032 BufferDesc::BufferDesc() + 12
Mon Jul 12 14:29:47.514 2010: 3:00000000400A0D64 BufferHashTab::new_entry() + 34
Mon Jul 12 14:29:47.513 2010: 4:0000000040051373 SharedHashTab::lookupAndHold(CacheObj**, ObjKey
const&, StripeGroup*, int) + 303
Mon Jul 12 14:29:47.514 2010: 5:0000000040049B43 HandleMBHashLookup(MBHashLookupParms*) + A3
Mon Jul 12 14:29:47.513 2010: 6:0000000040048BCE Mailbox::msgHandlerBody(void*) + 27E
Mon Jul 12 14:29:47.514 2010: 7:000000004003EEE0 Thread::callBody(Thread*) + A0
Mon Jul 12 14:29:47.513 2010: 8:000000004003BAF5 Thread::callBodyWrapper(Thread*) + A5
Mon Jul 12 14:29:47.514 2010: 9:00002AB47A8DD143 start_thread + 93
Mon Jul 12 14:29:47.553 2010: Signal 6 at location 0x2AB47AEB81F1 in process 884, link reg
0xFFFFFFFFFFFFFFFF.
```

```
Mon Jul 12 14:29:47.554 2010: mmfsd is shutting down.
```

Based on *Traceback* from the assert and the *insufficient memory* error from that application, the file system unmount points the investigation toward possible issues with maxFilesToCache and maxStatCache tunable parameters.

The **mmlsconfig** output shows that the mFTC and mSC values have been changed from the default of 1000 and 4000 respectively to 2000000 for each, as shown in Example 8-52.

Example 8-52 The mFTC and mSC values from the mmlsconfig command output

```
Configuration data for cluster mantest.usa:
-------------------------------------------
clusterName mantest.usa
clusterId 13882457470256956975
autoload no
autoload yes
maxFilesToCache 2000000
maxStatCache 2000000
minReleaseLevel 3.4.0.0
dmapiFileHandleSize 32
cipherList AUTHONLY
opensslibname /usr/lib/libssl.a(libssl64.so.0.9.8)
adminMode central
```

2. To determine whether the values for mFTC and mSC are tuned correctly for this cluster, determine whether the mmfs daemon can allocate all the mFTC and mSC that was requested. Look at the “dump fs” stanza of the **dump all** command that was previously collected or execute an **mmfsadm dump fs** command on any node and look for the `UMALLOC limits` heading. The `fileCacheLimit` and `statCacheLimit` lines are of interest. See Example 8-53.

Example 8-53 Checking for the tunable parameters fileCacheLimit and statCacheLimit

```
UMALLOC limits:
    bufferDescLimit     101855 desired   262144
    fileCacheLimit      777097 desired  2000000
    statCacheLimit      777097 desired  2000000
    diskAddrBuffLimit   155419 desired   400000
```

The `fileCacheLimit` and `statCacheLimit` lines show that the mmfs daemon can allocate only 777097 of the 2000000 mFTC and mSC that are requested. This indication means that the mFTC and mSC have been over committed for this cluster and are in need of tuning.

8.11.5 Scenario 5: Spectrum Scale file system not mounting

Usually when one or more Spectrum Scale file systems are not mounting, the following error, messages, or conditions can occur:

- While performing a manual mount of the file system, you get errors from either the operating system or Spectrum Scale.
- Your application cannot access the directory or file, and generate errors.
- The **mmlsmount** command indicates that the file system is not mounted on certain nodes.

When facing a situation where the Spectrum Scale file system is not mounting, you can follow these steps:

1. On a quorum node in the cluster that owns the file system, verify that quorum has been achieved. Check the Spectrum Scale log to see if an `mmfsd ready` message has been logged, and that no errors were reported on this or other nodes.
2. Verify that a conflicting command is not running. This applies only to the cluster that owns the file system. However, other clusters would be prevented from mounting the file system if a conflicting command is running in the cluster.

   For example, a **mount** command cannot be issued while the **mmfsck** command is running and it must not be issued until the conflicting command completes. It is not recommended to interrupt the **mmfsck** command because the file system will not be mountable until the **mmfsck** command completes. Try again after the conflicting command has completed.
3. Verify that sufficient disks are available to access the file system by issuing the **mmlsdisk** command. Note that Spectrum Scale requires a minimum number of disks to find a current copy of the core metadata. If sufficient disks cannot be accessed, the mount will fail. The corrective action is to fix the path to the disk.
4. Verify that communication paths to the other nodes are available. The lack of communication paths between all nodes in the cluster may impede contact with the file system manager.
5. Verify that the Spectrum Scale daemon on the file system manager is available. Run the **mmlsmgr** command to determine which node is currently assigned as the file system manager. If the Spectrum Scale deamon is not active, try to follow the steps described in “Scenario 1: Spectrum Scale daemon not running” on page 453.
6. Check to see if the mount point directory exists and that there is an entry for the file system in the `/etc/fstab` file (for Linux) or `/etc/filesystems` file (for AIX). If any of these elements are missing, an update to the configuration information may not have been propagated to this node. Issue the **mmrefresh** command to rebuild the configuration information on the node and reissue the **mmmount** command. Do not add Spectrum Scale file system information to `/etc/filesystems` (for AIX) or `/etc/fstab` (for Linux) manually. If after running **mmrefresh -f**, the file system information is still missing from `/etc/filesystems` (for AIX) or `/etc/fstab` (for Linux), contact the IBM Support Center.
7. Check the number of file systems that are already mounted. There is a maximum number of 256 mounted file systems for a Spectrum Scale cluster. Remote file systems are included in this number.
8. If you issue **mmchfs -V compat**, it enables backwardly compatible format changes only. Nodes in remote clusters that were able to mount the file system before will still be able to do so. If you issue **mmchfs -V full**, it enables all new functions that require different on-disk data structures. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that has the file system mounted at the time this command is issued, the **mmchfs** command will fail. All nodes that access the file system must be upgraded to the same level of GPFS or Spectrum Scale.

9. Issue the **mmlsfs -A** command to check whether the automatic mount option has been specified. If the automatic mount option is expected, check the Spectrum Scale log in the cluster that owns and serves the file system for progress reports indicating the following status:

```
starting ...
mounting ...
mounted ...
```

10. In some cases, the **mmfsck** command might be needed before mounting the file system. Usually this happens if the Spectrum Scale storage system were affected by some power outage or other problem that led to a power off. Before running the **mmfsck** command, run the **mmlsmount** command to ensure that the file system is not mounted in any node.

    See Example 8-54 for the **mmfsck** command output.

Example 8-54 The mmfsck command output

```
root@fsaix1:/usr/lpp/mmfs/bin>mmfsck /dev/wpargpfs01
Checking "wpargpfs01"
Checking reserved files
   5 % complete on Thu Oct 16 16:17:06 2014
Checking inodes
  45 % complete on Thu Oct 16 16:17:08 2014
Checking inode map file
  50 % complete on Thu Oct 16 16:17:10 2014
  52 % complete on Thu Oct 16 16:17:10 2014
  54 % complete on Thu Oct 16 16:17:10 2014
Checking ACL file records
  56 % complete on Thu Oct 16 16:17:10 2014
Checking directories and files
Checking log files
Checking extended attributes file
Checking allocation summary file
Checking policy file
Checking metadata of filesets
Checking file reference counts
  98 % complete on Thu Oct 16 16:17:11 2014
Checking file system replication status
 100 % complete on Thu Oct 16 16:17:11 2014

               65792 inodes
                  41   allocated
                   0   repairable
                   0   repaired
                   0   damaged
                   0   deallocated
                   0   orphaned
                   0   attached
                   0   corrupt ACL references

              655360 subblocks
               35771   allocated
                   0   unreferenced
                   0   duplicates
                   0   deletable
                   0   deallocated
```

```
                  992 addresses
                    0   suspended
                    0   duplicates
                    0   reserved file holes found
                    0   reserved file holes repaired
File system is clean.
```

11. Verify the `maxblocksize` configuration parameter on all clusters involved. If `maxblocksize` is less than the block size of the local or remote file system you are attempting to mount, you will not be able to mount it. To verify the `maxblocksize`, you can execute the following command:

    `mmfsconfig maxblocksize`

    The output should be similar with the following:

    `maxblocksize 1M`

    It can be changed by the **`mmchconfig`** command to the value that matches the correct *blocksize* used in the Spectrum Scale cluster.

8.11.6 Scenario 6: Upgrading GPFS or Spectrum Scale

When upgrading, consider what software you have upgraded, for example, upgrading Spectrum Scale from one PTF to another and going from one AIX Technical Level or Linux Service Pack to the next. In upgrades such as these, be sure to check the `mmfs.logs`, and **`errpt -a`** or /var/log/messages for issues with the OS, network, or disk subsystem that can affect the Spectrum Scale cluster. Pursue those issues either first or in parallel with the Spectrum Scale efforts.

A typical problem after upgrading Spectrum Scale from an earlier version is that the mmfsd daemon fails to start. The `/var/adm/mmfs.logs.latest` file indicates the error shown in Example 8-55.

Example 8-55 Checking the mmfs.log file for errors

```
Removing old /var/adm/ras/mmfs.log.* files:
Loading kernel extension from /usr/lpp/mmfs/bin . . .
GPFS: 6027-506 /usr/lpp/mmfs/bin/mmfskxload: /usr/lpp/mmfs/bin/aix64/mmfs64 is
already loaded at 66063360.
Sun Jul  4 02:54:15.924 2010: GPFS: 6027-1548 Error: daemon and kernel extension
do not match.
```

The problem is that the new code, either a PTF or version upgrade, was applied and the kernel extension from the previous PTF or version upgrade did not unload. The solution is to unload the old kernel extension so that the new one can be loaded.

This step can be accomplished in two ways:

- The best and most complete way is to reboot the node or nodes that are receiving this error. This way ensures that everything is loaded correctly. See Example 8-56.

Example 8-56 Rebooting the node to load the kernel extension

```
Sun Jul  4 04:03:12 MEDT 2010: Node rebooted.  Starting mmautoload...
Sun Jul  4 04:03:24 MEDT 2010: runmmfs starting
Removing old /var/adm/ras/mmfs.log.* files:
Loading kernel extension from /usr/lpp/mmfs/bin . . .
GPFS: 6027-500 /usr/lpp/mmfs/bin/aix64/mmfs64 loaded and configured.
```

```
Sun Jul  4 04:03:25.353 2010: GPFS: 6027-310 mmfsd64 initializing. {Version:
3.3.0.6   Built: May 27 2010 16:43:37} ...
```

- The second way is to run the **mmfsenv -u** command to unload the kernel extension. This command attempts to unload the kernel extension. For instance, if the command is successful on AIX, the message stating that the kernel extension has unloaded is displayed (Example 8-57). If unsuccessful, the kernel is still loaded (Example 8-58).

Example 8-57 Unloading the kernel extension

```
# mmfsenv -u
/usr/lpp/mmfs/bin/mmfskxunload: module /usr/lpp/mmfs/bin/aix32/mmfs unloaded.
```

If the kernel extension cannot be unloaded, rebooting is mandatory.

Example 8-58 The kernel is still loaded

```
# mmfsenv -u
sysconfig(SYS_CFGKMOD-term): Device busy
```

8.11.7 Scenario 7: NSD failures

Usually the Spectrum Scale NSD failures are generated by failures on the external storage subsystem. When this situation happens some of the following problems are observed:

- Your file system has been forced to unmount.
- The **mmlsmount** command indicates that the file system is not mounted on certain nodes.
- Your application is getting EIO errors.
- Operating system error logs indicate that you have stopped using a disk in a replicated system, but your replication continues to operate.
- The **mmlsdisk** command shows that disks are down.

Note: If you are reinstalling the operating system on one node and erasing all partitions from the system, Spectrum Scale descriptors are removed from any NSD that this node can access locally. The results of this action might require re-creating the file system and restoring from backup. If you experience this problem, do not unmount the file system on any node that is currently mounting the file system. Contact the IBM Support Center immediately to see if the problem can be corrected.

Example 1

In this example, an error was generated while creating NSD disks.

To create the NSDs, the Spectrum Scale requires physical disk devices access. This is done using the **mmcrnsd** command.

For disks that are SAN-attached to all nodes in the cluster, `device=DiskName` should refer to the disk device name in `/dev` on the node where the **mmcrnsd** command is issued. If a server list is specified, `device=DiskName` must refer to the name of the disk on the first server node. The same disk can have different local names on different nodes.

When you specify an NSD server node, that node performs all disk I/O operations on behalf of nodes in the cluster that do not have connectivity to the disk. You can also specify up to

eight additional NSD server nodes. These additional NSD servers will become active if the first NSD server node fails or is unavailable.

When the **mmcrnsd** command encounters an error condition, one of these messages is displayed:

6027-2108
Error found while processing stanza
or
6027-1636
Error found while checking disk descriptor descriptor

Usually, this message is preceded by one or more messages describing the error more specifically.

Another possible error from **mmcrnsd** is:

6027-2109 Failed while processing disk stanza on node nodeName.
or
6027-1661 Failed while processing disk descriptor descriptor on node nodeName.

One of these errors can occur if an NSD server node does not have read and write access to the disk. The NSD server node needs to write an NSD volume ID to the raw disk. If an additional NSD server node is specified, that NSD server node will scan its disks to find this NSD volume ID string. If the disk is SAN-attached to all nodes in the cluster, the NSD volume ID is written to the disk by the node on which the **mmcrnsd** command is running.

Example 2

In this example, the Spectrum Scale has declared NSDs as down.

There are several situations in which disks can appear to fail to Spectrum Scale. Almost all of these situations involve a failure of the underlying disk subsystem. The following information describes how Spectrum Scale reacts to these failures and how to find the cause.

Spectrum Scale will stop using a disk that is determined to have failed. This event is marked as MMFS_DISKFAIL in an error log entry (check the “MMFS_DISKFAIL” on page 417). The state of a disk can be checked by issuing the **mmlsdisk** command.

The consequences of stopping disk usage depend on what is stored on the disk:

- Certain data blocks may be unavailable because the data residing on a stopped disk is not replicated.
- Certain data blocks may be unavailable because the controlling metadata resides on a stopped disk.
- In conjunction with other disks that have failed, all copies of critical data structures may be unavailable resulting in the unavailability of the entire file system.

The disk will remain unavailable until its status is explicitly changed through the **mmchdisk** command.

After that command is issued, any replicas that exist on the failed disk are updated before the disk is used.

Spectrum Scale can declare disks *down* for a number of reasons:

- If the first NSD server goes down and additional NSD servers were not assigned, or all of the additional NSD servers are also down and no local device access is available on the node, the disks are marked as stopped.
- A failure of an underlying disk subsystem may result in a similar marking of disks as stopped.
  - Issue the **mmlsnsd -M** command to verify the relationship between the NSDs and the physical disks as shown in Example 8-59.

Example 8-59 The mmlsnsd -M command from an AIX server

```
root@fsaix1:/usr/lpp/mmfs/bin>mmlsnsd -M
Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------
 aix_lv1      AC101499543D385A   /dev/gpfs_lv   fsaix1
 aix_lv1      AC101499543D385A   -              fsaix2 directly attached
 aix_lv1      AC101499543D385A   -              fsaix3 directly attached
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix1
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix2
 gpfs2nsd     AC1014995436CE37   /dev/hdisk9    fsaix3
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix1
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix2
 gpfs5nsd     AC10149B543849BC   /dev/hdisk2    fsaix3
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix1
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix2
 gpfs6nsd     AC10149B543849BD   /dev/hdisk3    fsaix3
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix1
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix2
 gpfs7nsd     AC10149B54384D09   /dev/hdisk5    fsaix3
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix1
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix2
 gpfs8nsd     AC10149954385007   /dev/hdisk4    fsaix3
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix1
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix2
 gpfs9nsd     AC10149B543857AD   /dev/hdisk7    fsaix3
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix1
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix2
 tie1         AC1014995438B30C   /dev/hdisk1    fsaix3
```

  - Issue the **mmlsdisk** command to verify the status of the disks in the file system as shown in Example 8-60.

Example 8-60 The mmlsdisk shows the disk as "suspended"

```
root@fsaix1:/usr/lpp/mmfs/bin>mmlsdisk /dev/gpfs2
disk         driver   sector     failure holds    holds storage
name         type       size       group metadata data  status availability pool
------------ -------- ------ ----------- -------- ----- ------------ ------------
gpfs7nsd     nsd       512        -1      yes      yes   suspended up    system
```

  - Issue the **mmchdisk** command with the **-a** option to start all stopped disks. If the disk is in "*suspended*" state, the **mmchdisk** can be executed to re-enable the disk as shown in Example 8-61 on page 471.

Example 8-61 Changing disk status with mmchdisk

```
root@fsaix1:/usr/lpp/mmfs/bin>mmchdisk gpfs2 resume -d gpfs7nsd
root@fsaix1:/usr/lpp/mmfs/bin>mmlsdisk /dev/gpfs2
disk         driver   sector     failure holds    holds storage
name         type       size       group metadata data  status availability pool
------------ ------ ----------- -------- ----- ------------ ------------ ------------
gpfs7nsd     nsd         512        -1    yes      yes   ready      up     system
```

- Disk failures should be accompanied by error log entries (see “The operating system error log facility”) for the failing disk. Spectrum Scale error log entries labeled MMFS_DISKFAIL will occur on the node detecting the error. This error log entry will contain the identifier of the failed disk. Follow the problem determination and repair actions specified in your disk vendor problem determination guide. After performing problem determination and repair, issue the **mmchdisk** command to bring the disk back up.

9

Encryption

This chapter provides an overview about encryption within IBM Spectrum Scale. You should be familiar with IBM Spectrum Scale basics, be able to create file systems and filesets, and have a basic understanding of the policy engine in order to understand how encryption is achieved in IBM Spectrum Scale.

This chapter includes a detailed step-by-step procedure on how to set up an environment for running an encrypted file system.

The following topics are presented in this chapter:

- Introduction to encryption with IBM Spectrum Scale
- How encryption is implemented in IBM Spectrum Scale
- Step-by-step setup procedure
- Working with encryption
- Implementing access control on a node base

© Copyright IBM Corp. 2015. All rights reserved.

9.1 Introduction to encryption with IBM Spectrum Scale

Since Spectrum Scale clusters can span global environments and consist of various disk technologies, there is a need to protect data against unauthorized access. With IBM Spectrum Scale 4.1, the encryption feature is available for use. How encryption is implemented in Spectrum Scale is based on the following facts:

- There are no requirements for special hardware.
- There is the possibility to cascade to different security levels (with keys) to generate fine granularity.
- Provides Federal Information Processing Standard (FIPS)-compliant encryption.

Protection of data *at rest* uses encryption to make data unreadable to an attacker who does not possess the necessary decryption keys. *At rest* means that the data in encrypted on disk, and is decrypted at some point in the Spectrum Scale layer on the way to the reader. This is not aimed to protect data before it reaches Spectrum Scale, or while the data resides in the Spectrum Scale page pool (memory). Using Spectrum Scale encryption helps secure the data while on disk.

Encryption goes hand-in-hand with secure data deletion. Given the realities of modern storage hardware, it is very difficult, and in some cases impossible (think SSD) to assure that a given piece of data can be truly deleted or overwritten, or otherwise made unavailable to an attacker.

This design does not attempt to actively protect against network traffic snooping. However, as a nice side effect of encrypting data blocks before leaving the node on their way to a logical block device or an NSD server, an attacker cannot make use of network traffic snooping in such a setting to breach data confidentiality. This being said, the proposed solution does not claim to defend against an active network attacker which would modify or craft its own packets (for example, to compromise another Spectrum Scale node or delete data belonging to other tenants). The use of a *cipherList* could help defend against an active network attacker. Although the list was created for a multicluster, it could be used in this context to secure the network traffic.

The most common use case to address with the encryption feature is that some customers (mostly government) recommend a FIPS 140-2 compliant solution. FIPS is a validation process for cryptographic modules related to hardware and software and stands for Federal Information Processing Standard. The implementation procedure out of this IBM Redbooks publication describes the steps taken to build an FIPS-compliant solution. The GSKit that ships with Spectrum Scale is FIPS certified and it is used by Spectrum Scale for all crypto-related functions.

This chapter helps to understand the setup procedure of an encryption capable Spectrum Scale environment, and provides common-use encryption cases. For more details and theoretical background information, see the *IBM Spectrum Scale v4.1: Advanced Administration Guide*, SC23-7032-01.

Note: Encryption is currently only supported on IBM AIX, Linux x86_64, and Linux ppc64. Encryption on Windows operating system is not yet supported as of the writing of this IBM Redbooks publication.

9.2 How encryption is implemented in IBM Spectrum Scale

Encryption inside IBM Spectrum Scale is made of three major components:

1. A Spectrum Scale cluster, running at least release 4.1.
2. A remote key manager (RKM) for providing access control to keys and certificates (at the time of writing this IBM Redbooks publication).

> **Note:** Refer to the FAQ for any updates on RKM, in particular if you start with a Spectrum Scale release level beyond 4.1.0.4.

3. Encryption rules by Spectrum Scale policies.

   Every file is encrypted and decrypted by its file encryption key (FEK). The file encryption key is stored in the extended attribute (EA) of the file instead of the metadata because metadata is not encrypted. The FEK is finally used for encrypt and decrypt the data on disk. The FEK is not accessible directly. The FEK is encrypted with a master encryption key (MEK). When the FEK is wrapped, up to 8 MEKs at a time can be used for doing the wrap. A FEK cannot be stored unwrapped, it has to wrap with at least one MEK, and can be wrapped with up to 8 MEKs at a time. So in order to access the FEK, you need all the master encryption keys (MEKs) that the FEK was wrapped up with. In addition, you can wrap an FEK up to eight times with any other specific combination of MEKs. We show in the next sections how it works in more details. So you can finally have up to eight different wrapped versions of the FEK for a file. For a better understanding, see Figure 9-1.

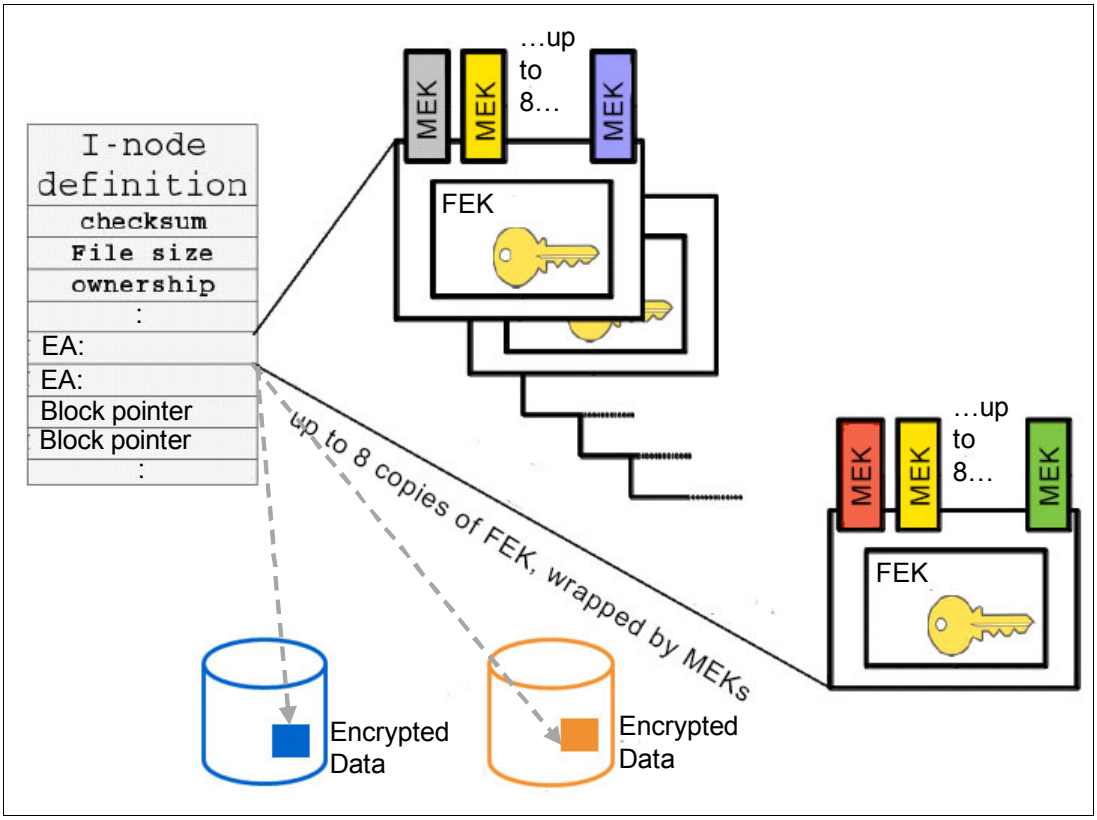


Figure 9-1 Mode information with MEK/FEK

At least one set of MEKs is needed to access the FEK. In the following sections, and complementing the examples presented, we show how you can wrap a rewrap FEK with more than one MEK. The MEKs are known by the GPFS daemon and read from an RKM.

The RKM might be any kind of key server. However, at the time of writing this publication, there is only one supported version for the RKM available: IBM Security Key Lifecycle Manager. We cover a detailed setup procedure of the key manager in 9.3.1, “Installing and setting up the IBM Security Key Lifecycle Manager” on page 477. When the IBM Security Lifecycle Manager is correctly set up, we show how Spectrum Scale uses it as an RKM. So, a given Spectrum Scale node gets the MEK from the RKM by authorizing itself with a certificate called *keystore*.

One reason why to separate the key management from the Spectrum Scale cluster is related to the fact that you can reach a higher security level. By managing the keys on a system outside of Spectrum Scale, you can separate the administrator’s rights accordingly and make sure that even super-user rights from inside the cluster cannot change the key configuration on the remote key management system. In addition, some helpful advantages go along with the managing keys on an IBM Security Lifecycle Manager including backup and restore of key data, making keys highly available.

The MEKs are stored remotely. Thus the way to get to these keys (MEK) is managed on a node based with certificates, which comes out of a node keystore. This means that a node needs to have a valid certificate (keystore) to get access to the MEK.

For now, we can summarize the flow, how to control the access to a file, on a node’s certificate which allows access to an RKM’s key and getting all the needed MEK to be able to unwrap and access the FEK.

The MEK is stored by Spectrum Scale in the cache of the nodes, thus there is no need to read the MEK on every file access.

Note: When an MEK is read and is known to the Spectrum Scale node, it remains in the Spectrum Scale daemon’s cache. Therefore, there is no performance penalty from holding the MEK remotely.

The decision if a file gets encrypted or not is given by the Spectrum Scale policy rules. Whether a file is encrypted is determined at file creation time by encryption-specific Spectrum Scale file placement policy rules. Up to eight rules can be applied.

The major goals for using encryption are to protect the data from unauthorized access. So if someone walks away with your disks, the data is absolutely unreadable.

IBM Spectrum Scale encryption may be seen as a way of access control to the data in the file system by keys. The keys are available on a node base. Therefore, encryption is not designed to use it for access control to files instead of using regular ownership and mode-bits inside the file system. This is an enhancement to grant a subset of nodes access to a subset of data. By having these key-based control mechanisms, secure deletion of data is addressed as well. This topic is described in 9.4.4, “Secure deletion file data” on page 500.

9.2.1 Support statement for encryption with other IBM Spectrum Scale features

With the introduction of IBM Spectrum Scale v4.1, encryption is supported within the local Spectrum Scale cluster and within multi-cluster environments. So encryption can be used to enforce further access control between remote cluster mounts as well. All known Spectrum

Scale features such as information lifecycle management (ILM), snapshots, **mmbackup**, and so on, work as usual with this encryption enhancement.

Note: IBM Spectrum Scale encryption is not supported on Windows operating system.

9.3 Step-by-step setup procedure

In this section, we explain the setup of an encrypted Spectrum Scale with an example. We create a cluster (BRAZIL) out of three nodes (RONALDO, NEYMAR, ROBINHO) and an IBM Security Lifecycle Manager (GERMANY). For an overview, refer to Figure 9-4 on page 481. The Spectrum Scale cluster does have a file system created. At this point, you should be able to set up a Spectrum Scale cluster. For details about how to set up an IBM Spectrum Scale cluster, refer to Chapter 2, “Infrastructure planning and considerations” on page 29 and Chapter 3, “Scenarios” on page 97.

9.3.1 Installing and setting up the IBM Security Key Lifecycle Manager

The IBM Security Key Lifecycle Manager: Installation and Configuration Guide, SC27-5335-01, provides detailed documentation about IBM Security Key Lifecycle Manager. However, this section gives a short documentation on its installation, which works for most common cases as it is a good starting point.

Getting the sources

IBM Security Key Lifecycle Manager is available for different platforms and operating systems. Depending on your platform, get the appropriate sources and prerequisites from:

The IBM Passport Advantage® website, which provides packages, is referred to as *eAssemblies*, for various IBM products:

http://www-01.ibm.com/software/passportadvantage/pao_customer.html

The IBM Fix Central website, which provides fixes and updates for software, hardware, and operating systems for your servers, can be found at this URL:

http://www-933.ibm.com/support/fixcentral

For UNIX based systems, the installation resources come as `.tar` files. Unpack the tar files to get a directory as shown in Example 9-1.

Example 9-1 Unpack installation resources

```
(nimres2/root) /nimrepo/TSKLM/linux_x86 > ll
total 8810656
-rw-r--r--    1 root     system   3826288640 Oct 13 16:43 SKLM_2.5_LIN64_1OF2_ML.TAR
-rw-r--r--    1 root     system    684759040 Oct 13 16:51 SKLM_2.5_LIN64_2OF2_ML.tar
drwxr-xr-x   11 root     system         4096 Oct 30 2013  disk1/
drwxr-xr-x    3 root     system          256 Oct 30 2013  disk2/
(nimres2/root) /nimrepo/TSKLM/linux_x86 >
```

For IBM AIX, you need more prerequisites to be able to start the installation procedure, which can be found at the following site:

ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/ezinstall/ppc/gtk2_bundle_v1.tar

In our example, we installed the IBM Security Lifecycle Manager on x86 Linux. For Linux, check that you have GTK, gtk2, and libXtst installed as shown in Example 9-2.

Example 9-2 Installing the operating system requisites

```
yum -y install glibc.i686
  yum install libgcc.i686
  yum install gtk2.i686
  yum install libXtst.i686
```

Minimum hardware and software requirements

Review Table 9-1 to verify the prerequisites for running IBM Security Key Lifecycle Manager v2.5.0.1.

Table 9-1 Installation prerequisites

System components	Minimum values	Suggested values
System memory (RAM)	4 GB	4 GB
Processor speed	Linux and Windows systems 3.0 GHz single processor AIX and Sun Solaris systems 1.5 GHz (2-way)	Linux and Windows systems 3.0 GHz dual processors AIX and Sun Solaris systems 1.5 GHz (4-way)
Disk space free for IBM Security Key Lifecycle Manager and prerequisite products such as DB2	5 GB	5 GB
Disk space free in `/tmp` or `C:\temp`	2 GB	2 GB
Disk space free in `/home` directory for DB2	5 GB	6 GB
Disk space free in `/var` directory for DB2	512 MB on Linux and UNIX operating systems	512 MB on Linux and UNIX operating systems

9.3.2 Installing IBM Security Lifecycle Manager

Now proceed to install IBM Security Key Lifecycle Manager. Change to the `disk1` directory and execute the install script as shown in Example 9-3. Check that you have a running x-environment and your X11 forwarding works.

Example 9-3 Starting the installation

```
[root@germany linux_x86]# ll
total 4405328
drwxr-xr-x. 11 root root       4096 Oct 30  2013 disk1
drwxr-xr-x.  3 root root        256 Oct 30  2013 disk2
-rw-r--r--.  1 root root 3826288640 Oct 13 16:43 SKLM_2.5_LIN64_1OF2_ML.TAR
-rw-r--r--.  1 root root  684759040 Oct 13 16:51 SKLM_2.5_LIN64_2OF2_ML.tar
[root@germany linux_x86]# cd disk1
[root@germany disk1]# ./install.sh
./install.sh
/mnt/linux_x86/disk1
/mnt/linux_x86/disk1
```

If some GTK or other packages for a correct run time environment are missing, you get an appropriate message in the terminal. During the installation, you see an X window installation wizard, which guides you through the installation. In Figure 9-2, we accept the default locations and continue.

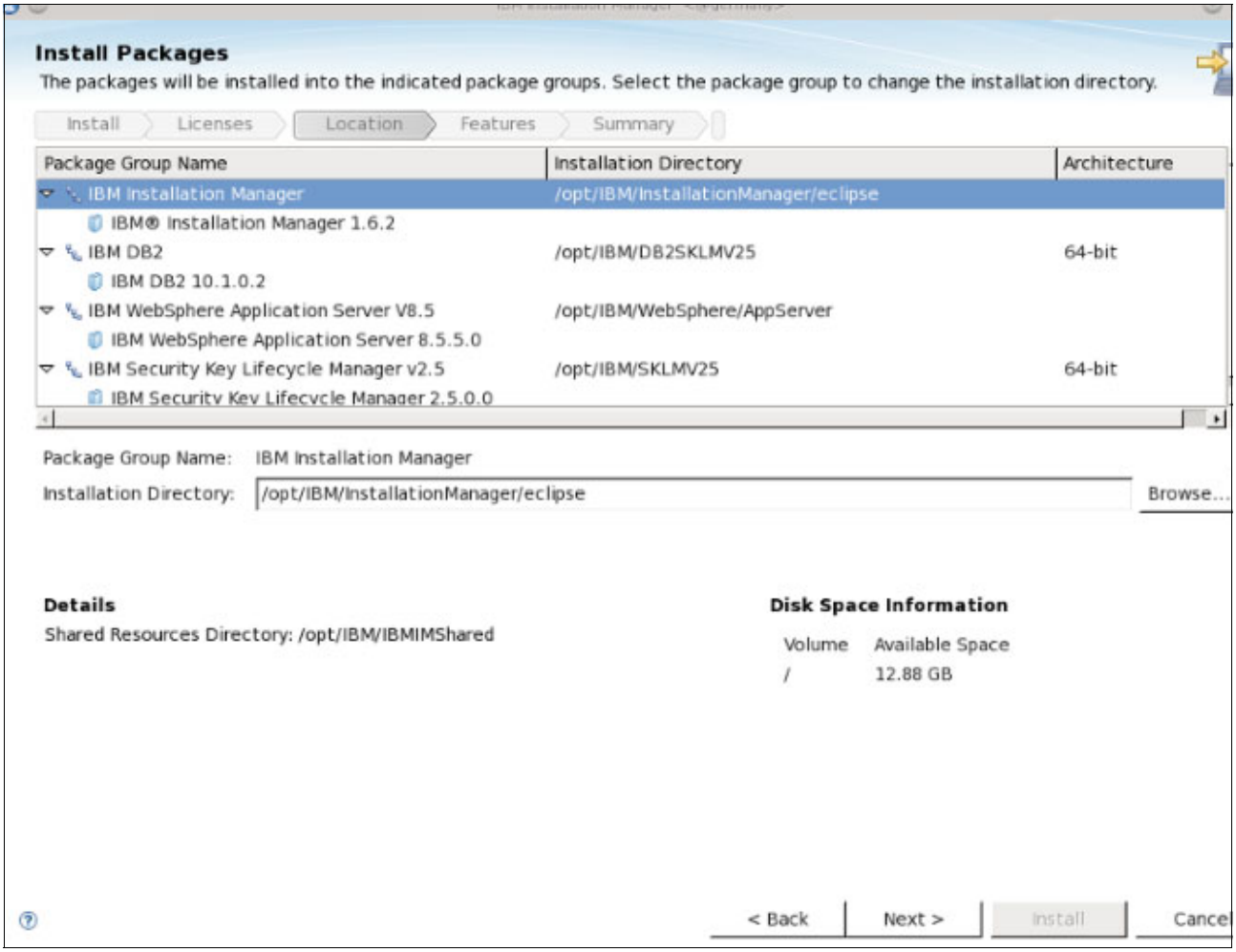


Figure 9-2 Installation wizard window

Continue to follow the installation procedures. Be careful in particular with your screen resolution as you might have to scroll to the left, otherwise you might miss necessary entry fields, but the wizard requires you to provide the information. Also, be careful again when completing the user and password information. Keep good notes about the user names and passwords.

Finally, the installation starts. Depending on your environment, the installation might take a while until you see the window as shown in Figure 9-3 on page 480.

Optionally, you can accept the defaults. Then, the wizard continues to create a working profile, creating the web service interface and WebSphere stanzas. At this time, we recommend selecting *none* and finish the installation here.

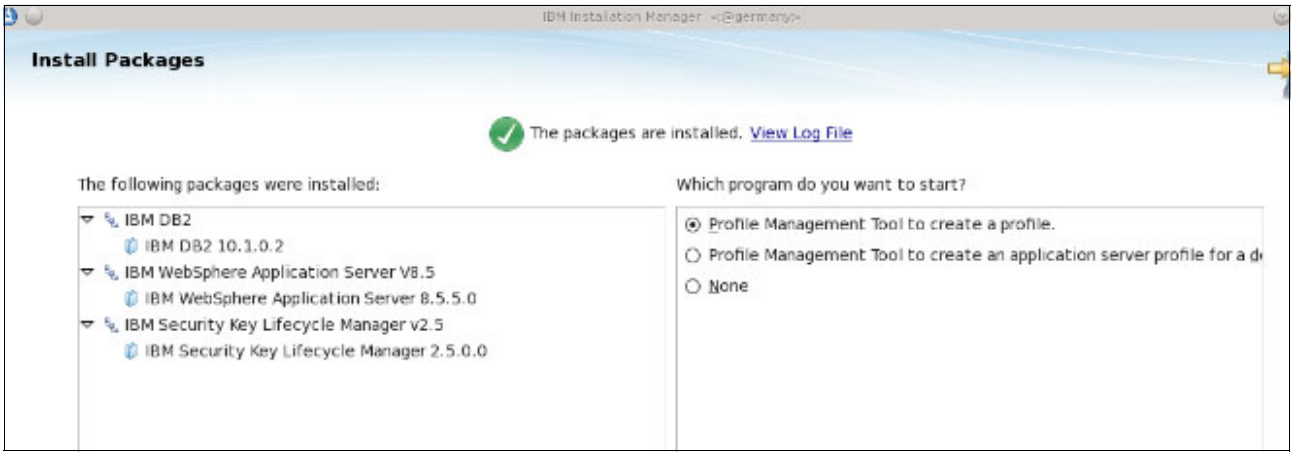

Install Packages
The packages are installed. View Log File
The following packages were installed:
IBM DB2
IBM DB2 10.1.0.2
IBM WebSphere Application Server V8.5
IBM WebSphere Application Server 8.5.5.0
IBM Security Key Lifecycle Manager v2.5
IBM Security Key Lifecycle Manager 2.5.0.0
Which program do you want to start?
Profile Management Tool to create a profile.
Profile Management Tool to create an application server profile for a d
None

Figure 9-3 Wizard finish window

9.3.3 Accessing KLM

You can access KLM with your browser. Keep in mind the firewall rules inside your key managers' operating system. The following ports need to be accessible from the outside: 9080, 3801, 441, and 5696. Also, keep in mind that you can change the default ports according to your needs. If you are not familiar with firewall rules, check your iptables to confirm that the application is running, thus the installation was successful.

You can access the KLM with:

`https://germany:9080/ibm/SKLM/jsp/Main.jsp`

We cover the configuration in the next sections.

Stopping and starting the key server

You can stop and start the server as written in Example 9-4. It might take some time after a node reboot for the services to be started.

Example 9-4 Starting and stopping KLM

```
[root@germany bin]# /opt/IBM/WebSphere/AppServer/bin/stopServer.sh server1
[...]
Username: wasadmin
Password:     # put your password here
 ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

## starting
root@germany bin]# /opt/IBM/WebSphere/AppServer/bin/startServer.sh server1
```

9.3.4 Updating the server

After a successful installation, go to IBM Fix Central, search for the *IBM Security Lifecycle Manager*, and download the latest fix pack for your operating environment. Follow the instructions on the `.readme` file within the same download directory.

Note: During update, RKM is not operable.

9.3.5 Creating certificates and keys for use with IBM Spectrum Scale and IBM Security Lifecycle Manager

When you have installed the RKM successfully, you now need to configure the RKM for further use with Spectrum Scale. Because key management can become very complex, we provide the steps in this section. Assume that there is a cluster named BRAZIL, and one external RKM named germany, as shown in Figure 9-4.

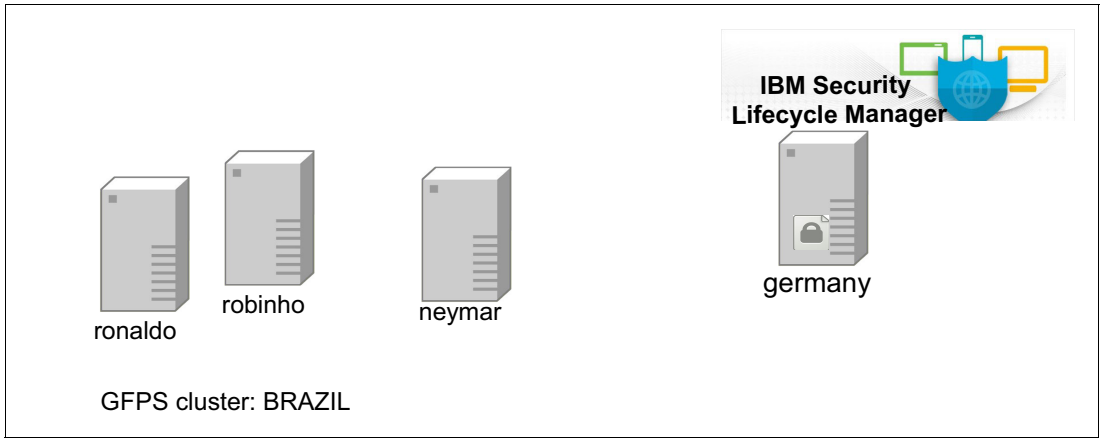


Figure 9-4 Overview for encryption setup

We need to configure the following (step-by-step):

- A server certificate (a running server can only have one valid server certificate).
- For the Spectrum Scale nodes, at least one pkcs12 keystore, containing the self-signed certificate.
- Distribute the client certificate to all members in the cluster.
  ** Optionally, additional unique client nodes certificates.
- Some keys on the RKM for later use by the clients.

The management of keys and the access control of Spectrum Scale nodes to the keys is all done on the RKM (germany) as shown in Figure 9-5 on page 482. However, the client certificates can be, and usually are, generated on the Spectrum Scale client nodes and imported to the RKM.

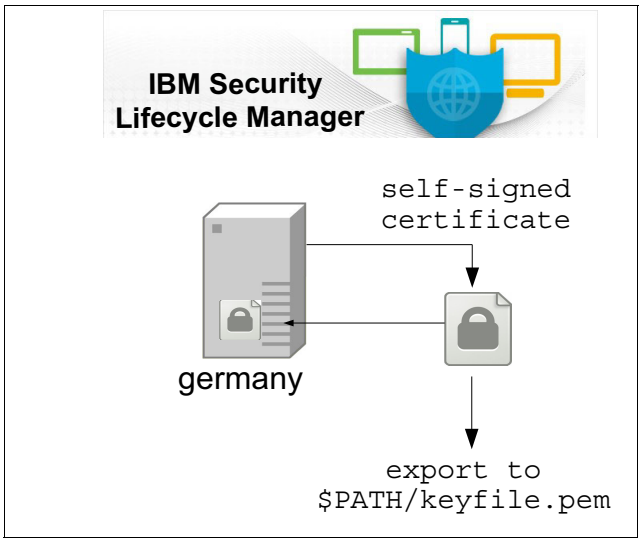


Figure 9-5 Management of the keys

Step 1: Create the server certificate

First, start by connecting to the RKM by using the web interface:

`https://germany:9080/ibm/SKLM/jsp/Main.jsp`

You now configure a server certificate. The server certificate is needed later by the clients to populate their keystore. To give an overview, see Figure 9-6 on page 483 to the left. As you can see, when the certificate on the server is created, you need to export it for further use on the client side.

To create the certificate over the GUI, click the Configuration tab and enter the appropriate names in the fields. In our example, we create a server certificate with the name *germany*. Optionally, you can provide a description. Refer to Figure 9-6 on page 483 for how to create the certificate.

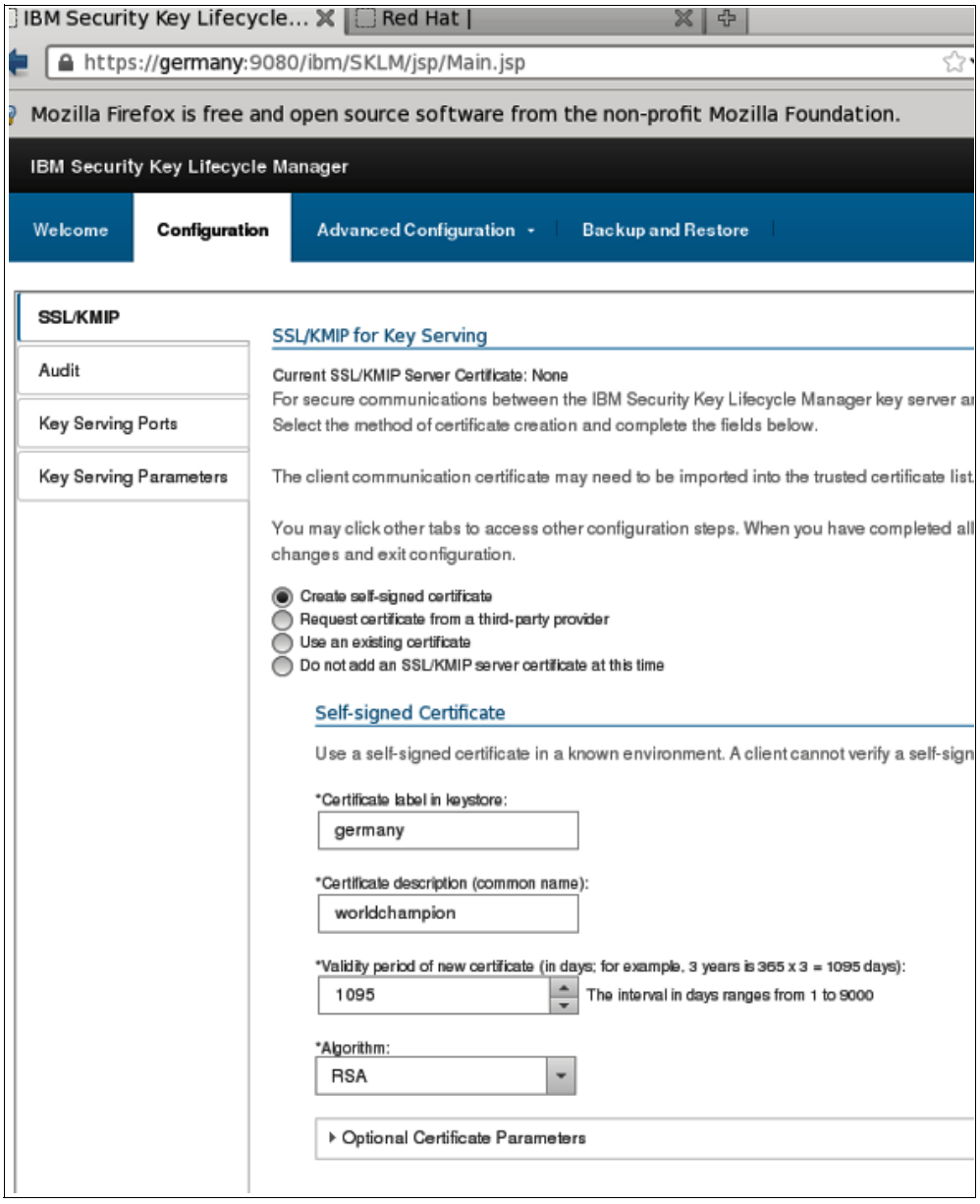

IBM Security Key Lifecycle...
Red Hat |
https://germany:9080/ibm/SKLM/jsp/Main.jsp
Mozilla Firefox is free and open source software from the non-profit Mozilla Foundation.
IBM Security Key Lifecycle Manager
Welcome
Configuration
Advanced Configuration
Backup and Restore
SSL/KMIP
Audit
Key Serving Ports
Key Serving Parameters
SSL/KMIP for Key Serving
Current SSL/KMIP Server Certificate: None
For secure communications between the IBM Security Key Lifecycle Manager key server a
Select the method of certificate creation and complete the fields below.
The client communication certificate may need to be imported into the trusted certificate list.
You may click other tabs to access other configuration steps. When you have completed all
changes and exit configuration.
Create self-signed certificate
Request certificate from a third-party provider
Use an existing certificate
Do not add an SSL/KMIP server certificate at this time
Self-signed Certificate
Use a self-signed certificate in a known environment. A client cannot verify a self-sign
*Certificate label in keystore:
germany
*Certificate description (common name):
worldchampion
*Validity period of new certificate (in days; for example, 3 years is 365 x 3 = 1095 days):
1095
The interval in days ranges from 1 to 9000
*Algorithm:
RSA
Optional Certificate Parameters

Figure 9-6 Creating the server certificate by IBM Security Lifecycle Manager

To check that the server can operate with your certificate, restart your server (see “Stopping and starting the key server” on page 480) and read in the key within the same menu. Select *use an existing certificate* as shown in Figure 9-7 on page 484 and enter your certificate again.

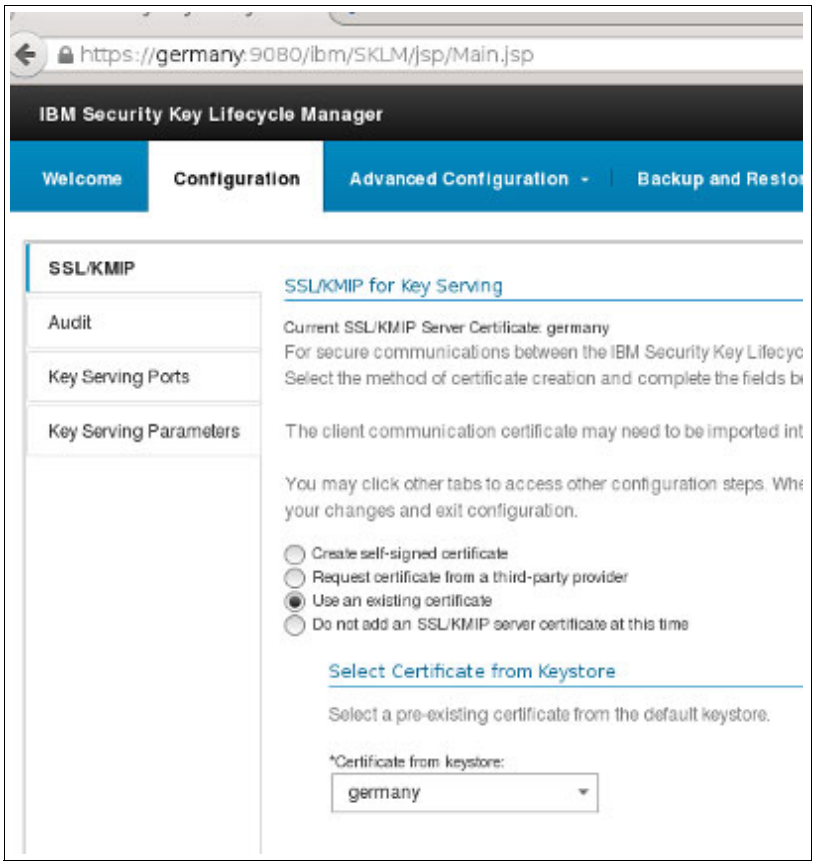


Figure 9-7 Activating the certificate on IBM Security Lifecycle Manager

After the activation step, your server certificate is ready to use. You can additionally verify that it is active now as shown in Figure 9-8.

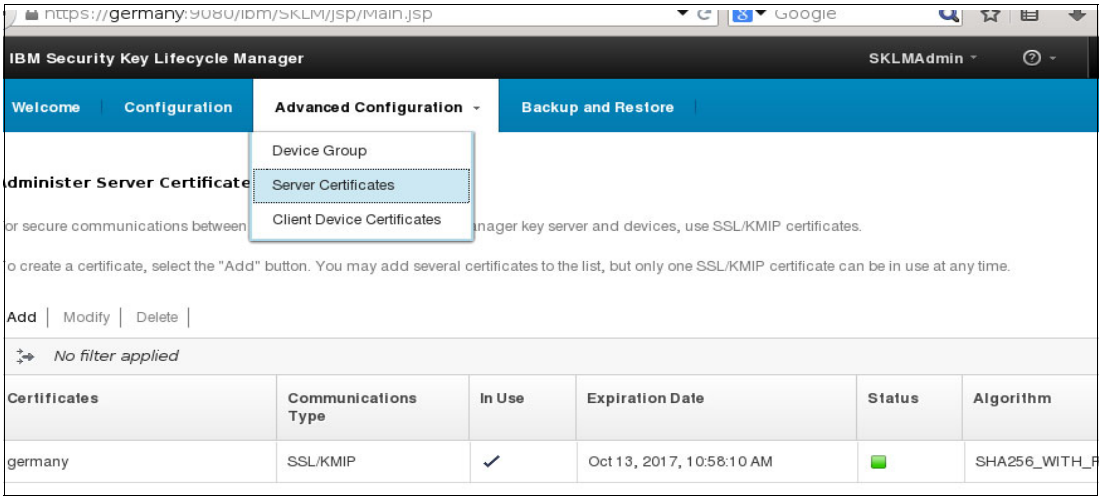


Figure 9-8 Verifying the server certificate

When the key is created, you need to export it into a file to share with the clients. To do so, switch to the command-line interface now as shown in Example 9-5 on page 485.

Example 9-5 Connect to the WebSphere Application Server console

```
ssh root@germany
Last login: Tue Oct 14 11:16:53 2014 from 172.16.254.6
[root@germany ~]# cd /opt/IBM/WebSphere/AppServer/bin/
[root@germany bin]# ./wsadmin.sh -lang jython

[Realm/Cell Name: <default>
Username: SKLMAdmin
Password:
 WASX7209I: Connected to process "server1" on node SKLMNode using SOAP
connector;  The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>
```

You are now connected to the RKM and can export the certificate. Before exporting the certificate, check Example 9-6, which explains how to show all existing keys.

Example 9-6 Extract and export the server certificate to a file (1/2)

```
wsadmin>print AdminTask.tklmCertList()

wsadmin>print AdminTask.tklmCertList()
CTGKM0001I Command succeeded.

uuid = CERTIFICATE-dd859885-2bc7-4f55-b1ef-2934201326ba
alias = germany
key store name = defaultKeyStore
key state = ACTIVE
issuer name = CN=worldchampion
subject name = CN=worldchampion
creation date = 10/14/14 10:58:10 AM Eastern Daylight Time
expiration date = 10/13/17 10:58:10 AM Eastern Daylight Time
serial number = 493003841518325
```

Now export the key into a file by using the command line as shown in Example 9-7.

Example 9-7 Extracting and exporting the server certificate to a file (2/2)

```
wsadmin>print AdminTask.tklmCertExport('[-uuid CERTIFICATE-dd859885-2bc7-4f55-b1ef-2934201326ba -format
base64 -fileName /gpfs/germany_serverkey.pem]')
CTGKM0001I Command succeeded.
/gpfs/germany_serverkey.pem
wsadmin>
```

Step 2: Create client certificates

The RKM accepts only certificates that are built with the server certificate. Therefore, we need the exported certificate file from the RKM to create the clients' keystore. In Example 9-8, it was a file named `germany_serverkey.pem`. So as the first step, you create a keystore on your Spectrum Scale node containing the server certificate. From an administrator console on the Spectrum Scale node, create the directory `RKMcerts` in `/var/mmfs/etc` and then execute the command as shown in Example 9-8.

Example 9-8 Generating client key store

```
mmauth gencert --cname cname_test --label client --cert /tmp/germany_serverkey.pem
--out /var/mmfs/etc/RKMcerts/client.p12 --pwd a_password
```

Replace the *cname_test* with a string identifying the Spectrum Scale node or the tenant-performing encryption. It is just a name, choose whatever you want to. Change *a_label* with a string that identifies the client certificate in the keystore (on the RKM) and change *a_password* with a password protecting the secret key material in the keystore file. Upon success, the new keystore has been created in `/var/mmfs/etc/RKMcerts/client.p12`.

Note: Use `/var/mmfs/etc/RKMcerts` for storing the certificates and keys.

The keystore with file name `client.p12` is distributed to every node in the cluster and to the same directory as shown in Example 9-9.

Example 9-9 Distributing the common client keystore and create individual certificates

```
[# for i in ronaldo robinho neymar
> do
> scp ./client.p12 $i:/var/mmfs/etc/RKMcerts/
> done
```

Optionally, we configure for every node an individual keystore, so that later, we can establish certain levels of access to files and distinguish between the nodes. Or if there is no need for differentiating the nodes, simply use the same client keystore on all nodes. For a quick approach on how to generate further keys, see Example 9-10 (some output is omitted to save space).

Example 9-10 Generating additional keystores (optional)

```
neymar:~  #mmauth gencert --cname neymar --label neymar --cert /tmp/germany_serverkey.pem --out
/var/mmfs/etc/RKMcerts/neymar.p12 --pwd beer
robinho:~ #mmauth gencert --cname robinho --label robinho --cert /tmp/germany_serverkey.pem --out
/var/mmfs/etc/RKMcerts/robinho.p12 --pwd wine
ronaldo:~ #mmauth gencert --cname ronaldo --label ronaldo --cert /tmp/germany_serverkey.pem --out
/var/mmfs/etc/RKMcerts/ronaldo.p12 --pwd water
```

Extracting the self-signed certificate from the pks12 file

At the time of writing this publication, the RKM does not trust unknown keystores in pkcs12 format. You need to tell the RKM that the RKM can trust the pkcs12 keystore with a corresponding certificate. So you have to extract this self-signed certificate out of the keystore. Refer to Example 9-11 for how to extract the certificate. As you can see, the file contains two certificates, a section for the server and a section for the client. You need the client only. The content of your file should look similar to Example 9-11. Do this step for every keystore, which is in the file in p12 format that you created in the previous step.

Example 9-11 Extracting the certificates out of the keystore

```
neymar:/var/mmfs/etc/RKMcerts # openssl pkcs12 -in neymar.p12 -nokeys -out neymar.crt
Enter Import Password: ## enter the password from Example 9-10
MAC verified OK



neymar:/var/mmfs/etc/RKMcerts # cat neymar.crt
Bag Attributes
    friendlyName: server
   [...]
issuer=/CN=germanies key
-----BEGIN CERTIFICATE-----
```

```
MIIC0TCCAbmgAwIBAgIGC8MTPXV6MA0GCSqGSIb3DQEBCwUAMBgxFjAUBgNVBAMT
[...]
-----END CERTIFICATE-----

Bag Attributes
    friendlyName: neymar
    localKeyID: 03 82 01 01 00 2D 90 3C 69 F1 F2 0F 05 BA 75 A5 AB 46 3D 70 7B D9 46 24 03 5F 78 A6 EF BD 09 67 74 39 78 E3 89 40 46 0A 09 17 01 2E CC 1F 1B E1 52
[...]
subject=/CN=neymar
issuer=/CN=neymar
-----BEGIN CERTIFICATE-----
[...] wmCPCDMys+4SbXvbNAjKxsKRZk2LtnKV5c=
-----END CERTIFICATE-----
```

To generate a certificate file, which only has the client part, simply edit the file and delete the server part. Use an editor of your choice. The final result should look like Example 9-12.

Example 9-12 Client certificate

```
neymar:/var/mmfs/etc/RKMcerts # cat neymar.crt
Bag Attributes
    friendlyName: neymar
    localKeyID: 03 82 01 01 00 2D 90 3C 69 F1 F2 0F 05 BA 75 A5 AB 46 3D 70 7B D9 46 24 03 5F 78 A6 EF BD 09
67 74 39 78 E3 89 40 46 0A 09 17 01 2E CC 1F 1B E1 52 F9 91 89 60 4F 60 30 17 B2 E0 BB 46 42 36 70 4D 8F D1 75
60 D4 AC E0 45 36 93 7F 19 3F 8B E7 43 00 0C B2 D3 A5 BA D2 2C 7F 6A A5 87 4E 8A 14 33 F5 B6 EF 96 AA E9 DE 99
6F 23 D5 06 6C 0F E4 76 14 BE 91 74 B7 CF 1D 04 61 F0 7F 01 01 82 EF 67 50 AE D5 BC 83 2F 15 F7 70 65 AD 54 50
69 B7 DB C8 D0 0E 59 8C 86 2D 75 8E FF 5B 9C 17 D9 C6 50 EC 61 BC 28 E0 BC EB 44 04 8C 57 A7 4B E9 82 8C 30 77
56 B3 0B EC 18 53 E0 A3 42 EA 7E E7 87 9F 7A CF E8 82 8E 1E 8F 4B 20 CE B7 BE 2C 15 5A E2 B2 17 4B AF 9A 36 47
E0 97 E8 BE 9F 95 CD 8B A3 AC 73 E2 ED BB D4 72 1D EB AD 1B 09 82 3C 20 CC CA CF B8 49 B5 EF 6C D0 23 2B 1B 0A
45 99 36 2E D9 CA 57 97
subject=/CN=neymar
issuer=/CN=neymar
-----BEGIN CERTIFICATE-----
MIIC5jCCAc6gAwIBAgIIMfAkxpt/DjYwDQYJKoZIhvcNAQELBQAwETEPMA0GA1UE
AxMGbmV5bWFyMB4XDTE0MTIxNDE5MjI0M1oXDTE3MTIxNDE5MjI0M1owETEPMA0G
A1UEAxMGbmV5bWFyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoPH4
TSqvvDLfaolKDaQ9oRGGdwVERIqWYmWEYR0FWateXK6r3UMK9Sc9w6BJgekA8zg9
dWDnVrMB9QseXl/lT8Wh9/sPggpi3SlZj4vQsxNEv0OZ/FCqG5Dbic59af7B6GXi
MCIYFKVzQFG8Kw9kNn+dJpU6HThWyDXUZpe6WTv1jR46
3ij1y++Bva90U8KEkNVK
RtcIdkvIbLlRXDooiTyuw3kvkKj2Dy8rv+hddweAbzJLF+U5LhPHpxttHiiTN9/7
8Q7vMahlk2MbgaX4YKx9PM89ghXH15CfRgpBPG6NDWGdVWj+nM4rn9i3qI5eeBm2
twqWfwGKlyBtJ+P8uwIDAQABo0IwQDAdBgNVHQ4EFgQUzs/ncoBND6SoQB3SUDYP
UO7a9M8wHwYDVR0jBBgwFoAUzs/ncoBND6SoQB3SUDYPUO7a9M8wDQYJKoZIhvcN
AQELBQADggEBAC2QPGnx8g8FunWlq0Y9cHvZRiQDX3im770JZ3Q5eOOJQEYKCRcB
LswfG+FS+ZGJYE9gMBey4LtGQjZwTY/RdWDUrOBFNpN/GT+L50MADLLTpbrSLH9q
pYdOihQz9bbvlqrp3plvI9UGbA/kdhS+kXS3zx0EYfB/AQGC72dQrtW8gy8V93Bl
rVRQabfbyNAOWYyGLXWO/1ucF9nGUOxhvCjgvOtEBIxXp0vpgowwd1azC+wYU+Cj
Qup+54efes/ogo4ej0sgzre+LBVa4rIXS6+aNkfgl+i+n5XNi6Osc+Ltu9RyHeut
GwmCPCDMys+4SbXvbNAjKxsKRZk2LtnKV5c=
-----END CERTIFICATE-----
neymar:/var/mmfs/etc/RKMcerts #
```

Do the same step for all keystores that you created before. You should finally have pairs of a p12 formatted keystore and a corresponding certificate out of this keystore. On the Spectrum Scale side, you just need the keystore files, formatted in pkcs12 format.

On the RKM, you need the corresponding certificates to finish the configuration. This is covered in the next section.

Summary

So finally, all the cluster nodes have the following keys (refer to Figure 9-9 on page 489 and the output from Example 9-13): One common client certificate, which is known by every cluster member, and an optional dedicated certificate unique for each node.

For a minimal configuration, one common client certificate would be enough. We just enhance the scenario in this book with more keystores for demonstration purposes.

For further configuration on the RKM, transfer the certificates to the RKM by remote copy: **ftp** or any other file transfer mechanism. The Spectrum Scale node needs the keystore in p12 format only. Distribute the common keystore (here client.p12) to every node and place it in the `/var/mmfs/etc/RKMcerts` directory.

Example 9-13 Client certificates

```
[root@ronaldo RKMcerts]# mmlscluster
[..]
  GPFS cluster name:         brazil.pele
  […]
 Node  Daemon node name  IP address     Admin node name  Designation
-------------------------------------------------------------------------
   2   neymar            172.16.20.162  neymar           quorum
   3   ronaldo           172.16.20.161  ronaldo          quorum
   4   robinho           172.16.20.163  robinho

[root@ronaldo RKMcerts]# mmdsh ls -l /var/mmfs/etc/RKMcerts/
ronaldo:  total 8
ronaldo:  -rw-r--r--. 1 root root 3306 Oct 17 09:51 client.p12
ronaldo:  -rw-r--r--. 1 root root 3251 Oct 17 16:55 ronaldo.p12
ronaldo:  -rw-r--r--. 1 root root 1962 Oct 17 16:55 ronaldo.crt
neymar:  total 8
neymar:  -rw-r--r--. 1 root root 3306 Oct 14 15:42 client.p12
neymar:  -rw-r--r--. 1 root root 1961 Oct 14 15:43 client.crt
neymar:  -rw-r--r--. 1 root root 3226 Oct 14 15:50 neymar.p12
neymar:  -rw-r--r--. 1 root root 1962 Oct 14 15:51 neymar.crt
robinho:  total 8
robinho:  -rw-r--r--. 1 root root 3306 Oct 17 09:51 client.p12
robinho:  -rw-r--r--. 1 root root 3252 Oct 17 09:55 robinho.p12
robinho:  -rw-r--r--. 1 root root 1961 Oct 17 09:56 robinho.crt
[root@ronaldo RKMcerts]#
```

Now we have the following situation as shown in Figure 9-9 on page 489. We have a valid server certificate and we have (minimum configuration) one common client keystore, which is shared among all nodes. Additionally, we have some individual client keystores and its corresponding certificates as well.

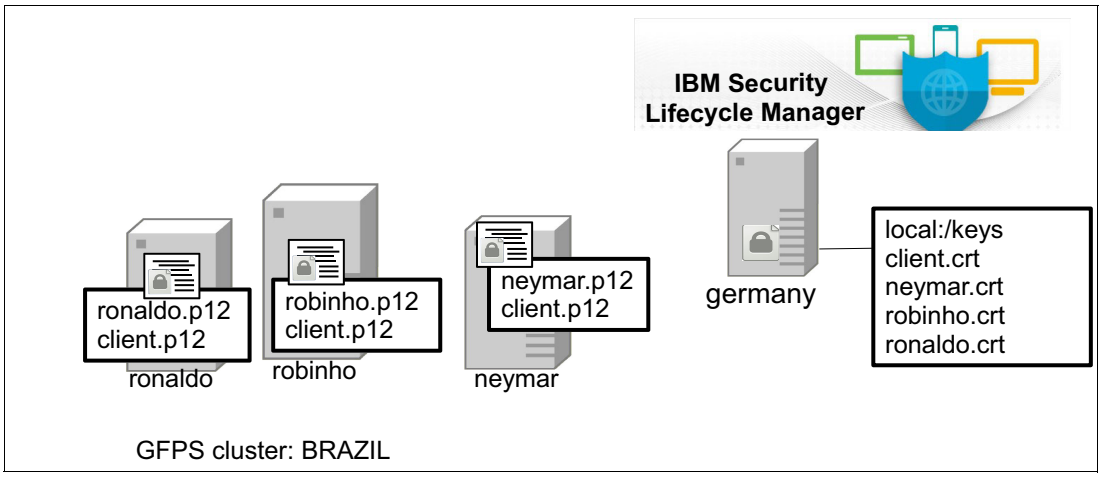


Figure 9-9 Certification configuration

9.3.6 Prepare IBM Security Lifecycle Manager for use with IBM Spectrum Scale

Now, once you have your keys and certificates, you can configure IBM Security Lifecycle Manager. It is much easier to configure IBM Security Lifecycle Manager if you already have a common understanding on how Spectrum Scale interacts with the RKM, and how encryption is brought to the files. We start with the RKM first.

We recommend reading the next paragraphs first, before you continue configuring your cluster. We now create a relationship, which nodes, identified by its certificates, can get access to which master keys.

Access the IBM Security Lifecycle Manager for administration

Enter the corresponding URL into your browser:

`https://germany:9080/ibm/SKLM/jsp/Main.jsp`

Then enter your userID and password. Remember that the default user is *SKLMadmin*. After successful login, you should get a WEB GUI as shown in Figure 9-10 on page 490.

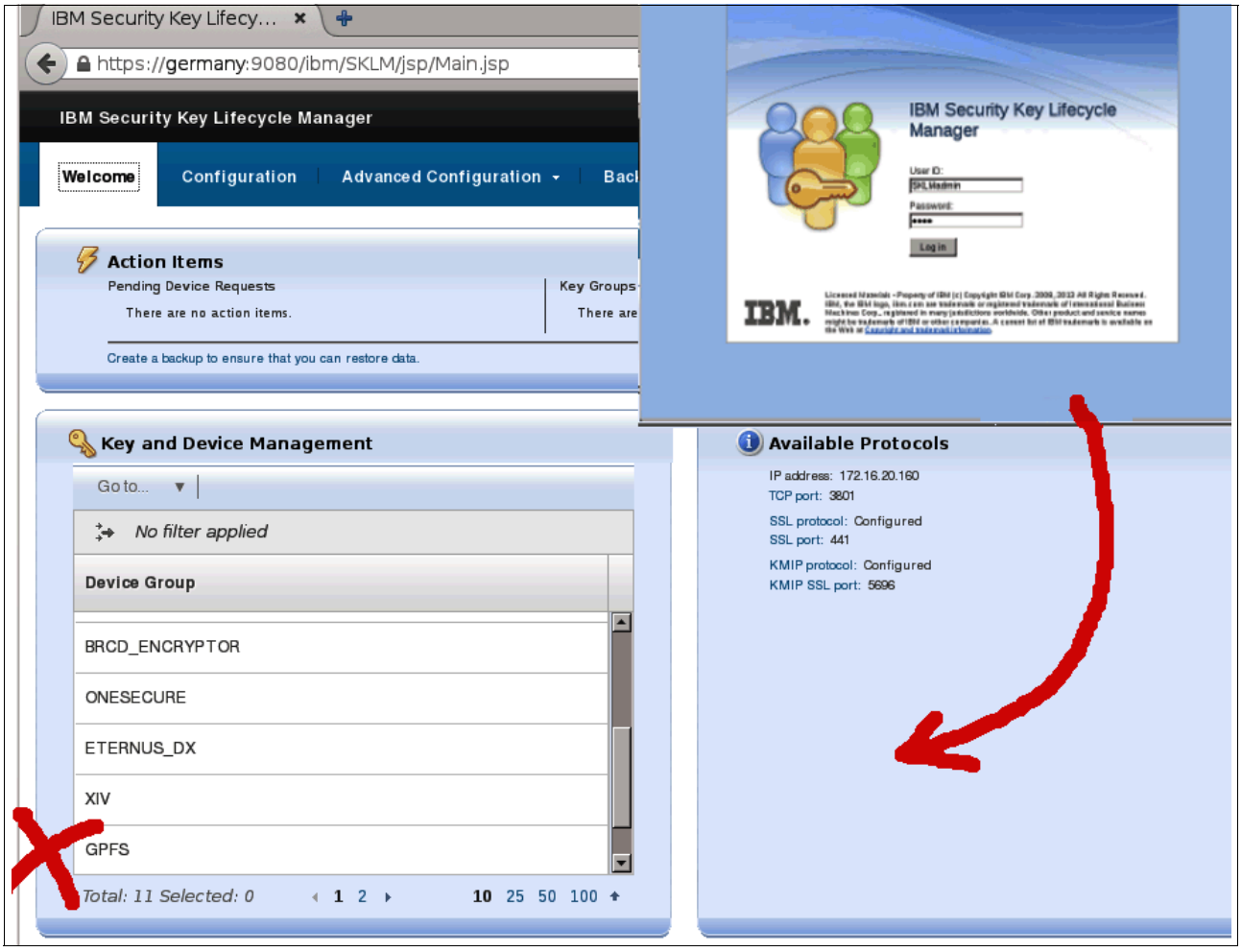


Figure 9-10 WEB GUI: Welcome page

Create the device group

By default, a device group is already configured which is called *GPFS*. We prefer to create our own device group (DG). A DG simply stands for a group of nodes that get access to the same keys. So as a good starting point, we create a DG group, where we later put in all the certificates of the nodes out of the cluster.

We now create a DG label GPFS_BRAZIL. Later we need this name for the appropriate Spectrum Scale configuration. Figure 9-11 on page 491 shows how to proceed to create your DG.

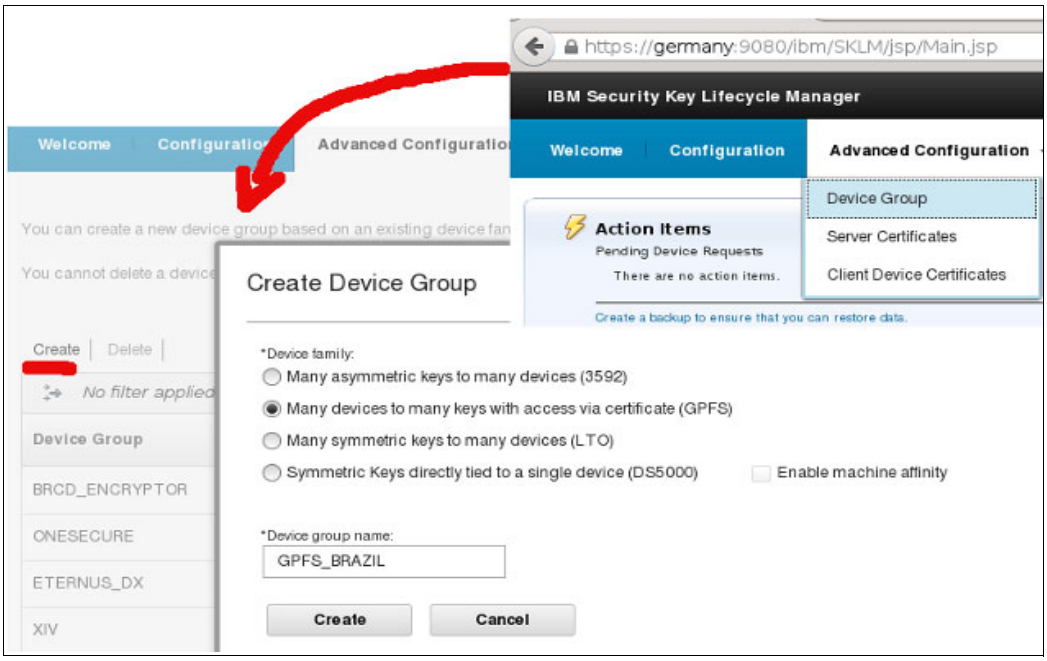


Figure 9-11 Creating a device group (DG)

Redo this step to have several DGs ready for further use. In your example, we created device groups with the name GPFS_BRAZIL and a second DG named GPFS_BRAZIL_HALF. We need them later for demonstration purposes.

If you just want to achieve a simple Spectrum Scale encryption and have all the files encrypted by the same key, you can skip the creation of the second DG.

Organizing the client's certificate and keys in the device group

First, generate the keys on the RKM. These keys are called MEKs from the Spectrum Scale perspective.

On the Welcome tab, highlight **DG** → **Go to** → **Manage Keys and certificates**. A dialog box opens as shown in Figure 9-12 on page 492 which helps create the keys. These keys are used later by Spectrum Scale as MEKs.

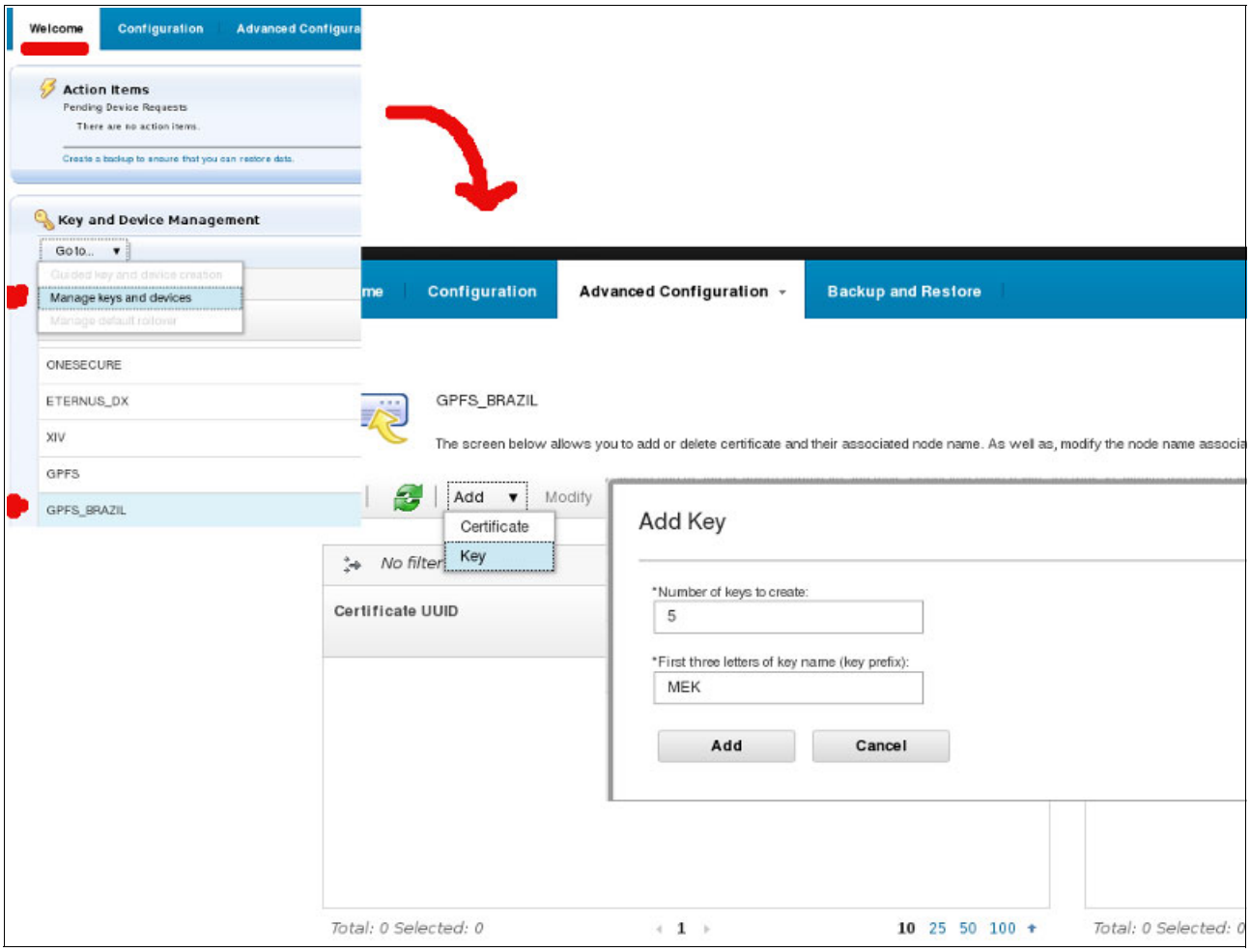


Figure 9-12 Creating the MEK

You can specify as many keys as you want to and at least you should create one key. In our example, we create five keys, and in the following sections we explain the reason for creating that many keys. At this point, you have created this key, which becomes one of the MEKs, and you need it later from the Spectrum Scale configuration side. You see a key as a pair of name and Key UUID (Figure 9-13). They key UUID is what you need in one of the next steps, when you configure the Spectrum Scale rules.

No filter applied

Key UUID	Name
KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678	mek000000000000000000
KEY-d3fd0266-b723-48be-b2db-8b52aa678c60	mek000000000000000001

Figure 9-13 UUID of an MEK

The next step is to configure who (client nodes) is allowed to access these keys. This is done by putting the appropriate client's certificate (which comes out of the keystore) into the device group. Before you can do this, you need to copy the client's certificates to the RKM master.

The certificates have to be available on the RKM. Remember where you store these files in your RKM server. We place all certificates under `/keys` in the RKM. Once they are locally available to the RKM, use the GUI, and put them into the configuration of the key manager as follows.

Go again on the *welcome* tab and highlight the *Device Group* again. Select *<Go to>* and *<Manage Keys and certificates>* to open the *Advanced configuration* view of the DG. This time, select certificate to add. Remember the local share where you stored the client's certificates. In our example, we put every client certificate into a directory called `/keys` on the RKM. Select your client keystore and complete the name for the certificate. Remember the name that you specified when creating the keystore. The name is given with the command-line option **`-name`** in the **`mmauth gencert`** command (see Example 9-8 on page 485).

So all Spectrum Scale nodes having the appropriate keystore available have access to all the MEKs out of the device group as shown in Figure 9-14.

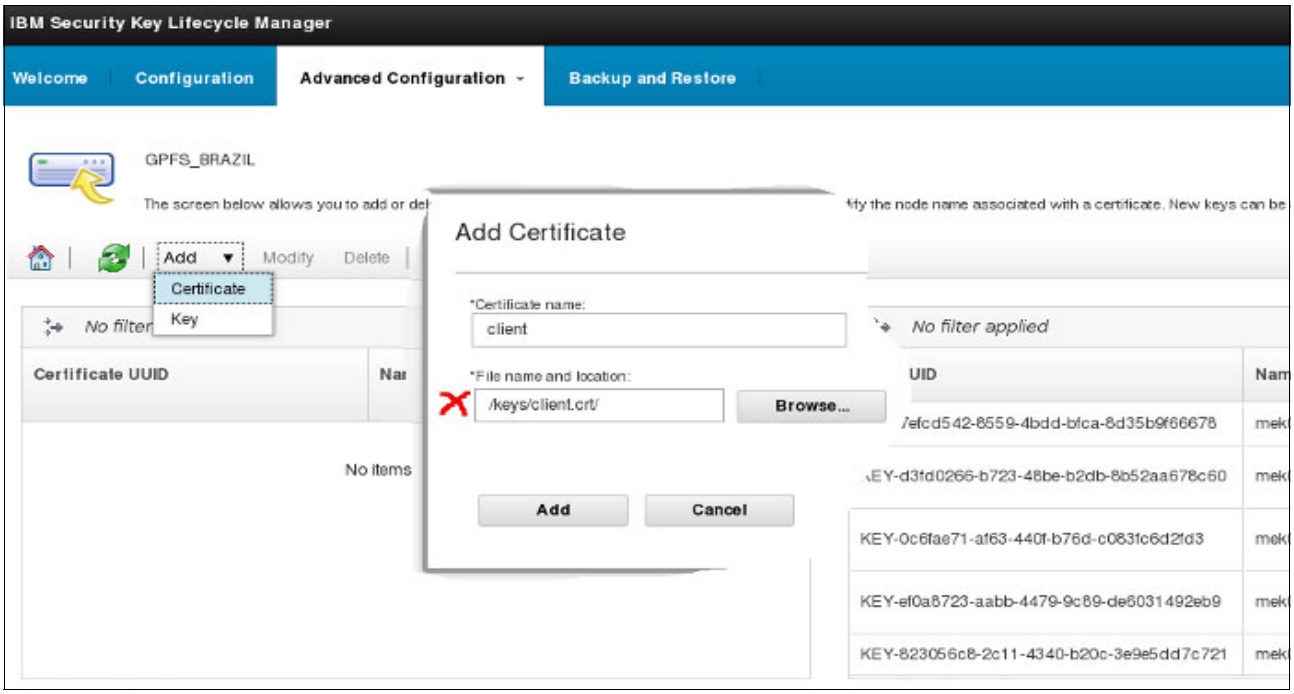


Figure 9-14 Adding client's certificates to DG

9.3.7 Configure IBM Spectrum Scale for encryption

Encryption in Spectrum Scale is done by policy rules using the benefit from the powerful search and select capabilities of the Spectrum Scale's policy engine. Encryption in Spectrum Scale is configured by placement rules. You have to provide at least one placement rule. You can configure Spectrum Scale to encrypt a certain subset of files, selected by name, filesets and many other attributes. All encryption rules matching to your pattern, applies to a file. You can specify as many encryption rules as you want, but your policy file must not exceed 1 MB and not more than eight encryption rules can be applied to a single file. However, for placing the file in the appropriate Spectrum Scale storage pool, still only one (the first matching) rule applies. Remember to specify a default placement rule for all others at the end of your policy file. We explain it in the next few sections.

Preparing Spectrum Scale

The Spectrum Scale cluster should be created and run with the latest available code level. At the time of writing this publication, we are at Spectrum Scale 4.1.0.4. The minimum level for support starts with 3.5.0.7. With the last recent code level, encryption is already enabled by default, once you have the appropriate Spectrum Scale rpms installed. For installing the crypto package to an existing cluster node, the mmfs daemon needs to be down.

Note: To enable Spectrum Scale clusters to run encryption, you need to install the gpfs.crypto package.

If you start with earlier releases, you have to check that the cluster configuration has the appropriate parameter for encryption enabled. To do so, check the configuration parameter **`encryptionenabled=yes`** with the **`mmlsconfig`** command or enable it with the **`mmchconfig`** command.

Note: The file system must have the attribute set for the fast extended attribute (*--fastea*).

For storing the keys in the EAs of the files, it is highly recommended having a file system with 4 K inode size created. You can run encryption on a file system with the standard inode size; however, depending on your encryption rules, especially when you wrap your FEK with several MEKs, the inode space for the EA might be less and your encryption can fail with I/O errors. If you intend to use existing file systems and migrating them to the latest Spectrum Scale level, you might consider re-creating the file system with the actual attributes.

Note: Following are the required attributes for the file system:

- Use a file system with a 4 K inode size.
- Enable fast EAs.
- The daemon version should be at the most recent code level.

To keep track in our example, we already created a cluster, which has the *gpfs.crypto* package installed. We call the cluster BRAZIL. It contains three nodes. Upon these nodes, a file system is created matching the preceding attributes. See Example 9-14 to verify the attributes.

Example 9-14 Minimum file system version and attributes

```
[root@ronaldo ~]# mmlscluster

GPFS cluster information
========================
[....] some output repressed for better overview [...]

 Node  Daemon node name  IP address     Admin node name  Designation
--------------------------------------------------------------------
   2   neymar           172.16.20.162  neymar           quorum
   3   ronaldo          172.16.20.161  ronaldo          quorum
   4   robinho          172.16.20.163  robinho

[root@ronaldo ~]# mmlsfs germany  -V --fastea --encryption -i -T
flag               value                    description
------------------ ------------------------ -----------------------------------
 --fastea          Yes                      Fast external attributes enabled?
 --encryption      Yes                      Encryption enabled?
 -V                14.10 (4.1.0.10)         File system version
 -i                4096                     Inode size in bytes
```

```
 -T                /gpfs/germanies_field    Default mount point
[root@ronaldo ~]#
```

The information about how Spectrum Scale decides if a file gets encrypted or not is in the rule. If an encryption rule applies to a file, further information comes out of the rule including which keys to use. The access path to that key, requested out of the rule, is in the configuration file `RKM.conf`. With the data out of `RKM.conf`, the nodes try to access the key. In fact, the node's mmfsd already has this key in cache once the policy is enabled. This is explained in more details in 9.4, "Working with encryption" on page 498. In case of success (getting or having the key), the file's FEK can be unwrapped and the file can be accessed by this node. See the flowchart in Figure 9-15 to get an overview about the next two paragraphs, where we describe the steps that you need to take to finish the encryption configuration.

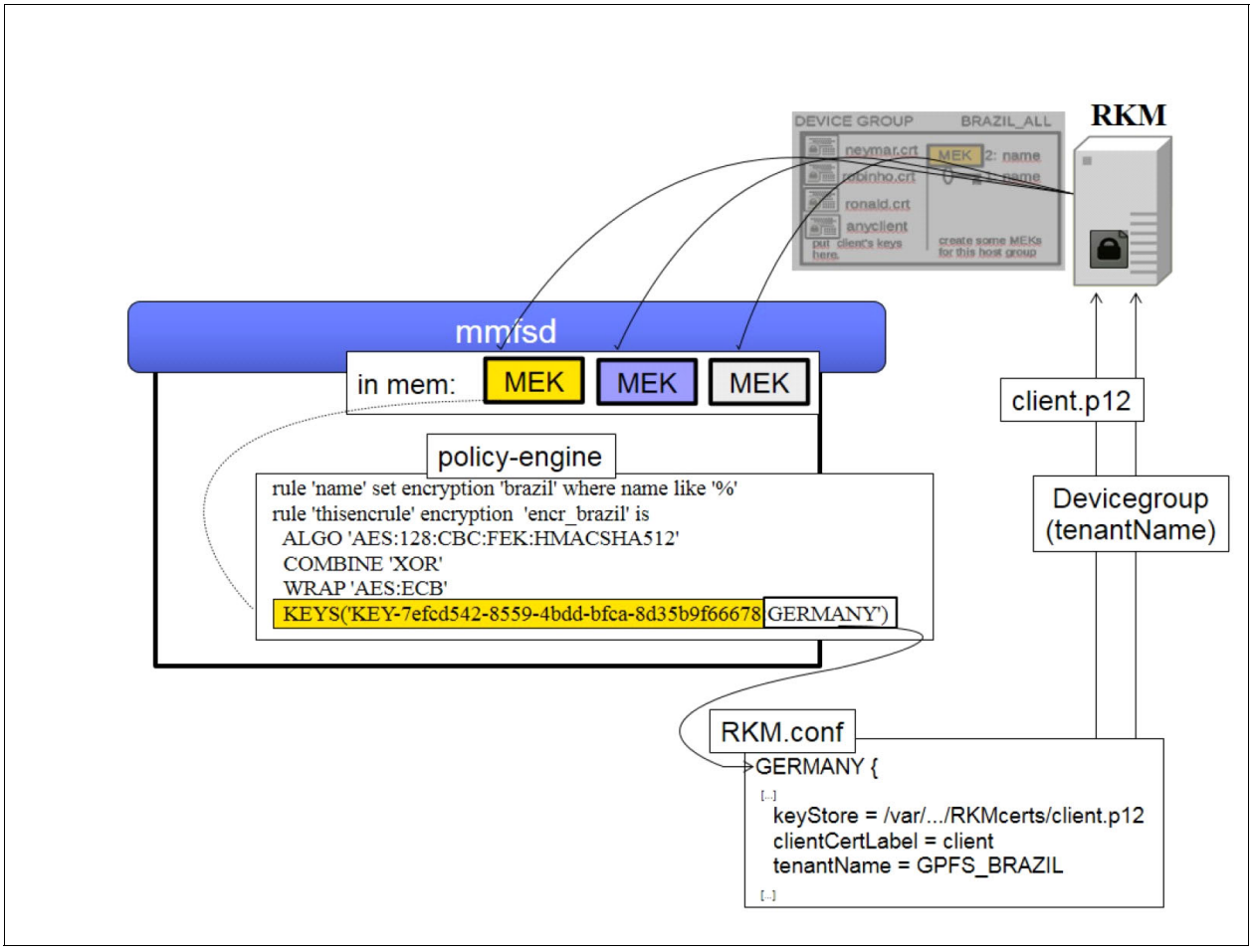


Figure 9-15 Encryption flow in Spectrum Scale

RKM.conf

The information about the location of the keys is stored in the `RKM.conf` file. Usually, the `RKM.conf` file can be identical on all nodes in the cluster. However, the `RKM.conf` file can be different or even unique to a node, if you are going to achieve different access levels between your nodes within the local or remote cluster. A sample configuration and quick description are given in Example 9-15 on page 496.

Example 9-15 Minimal RKM.conf sample

```
# cat /var/mmfs/etc/RKM.conf
GERMANY {
  type = ISKLM                                  # RKM is a ISKLM server
   kmipServerUri = tls://germany:5696           # TLS-connection to host on port
   keyStore = /var/mmfs/etc/RKMcerts/client.p12 # The path to the PKCS12 file containing server certificate
   passphrase = beer                            # Passphrase of private key
   clientCertLabel = client                     # Label of the client keypair to be used among those in the
                                                # keystore
   tenantName = GPFS_BRAZIL                     # Name of tenant (DG), set in ISKLM set-up
}
```

Your `RKM.conf` file can contain more than one entry. Every stanza has to begin with a name, which is called the RKMID. We need this RKMID later, when defining the policy rule. Within your RKMID stanza, check that the given client certificates are really available and accessible on that node and the label pass-phrase and PATH are correct.

While every node in your cluster shares the same key, you can easily distribute the `RKM.conf` among your nodes. If a node does not have all the client certificates referenced in the `RKM.conf` file, Spectrum Scale complains and you get several error messages.

Note: You can configure as many stanzas in the file as you might need, however the `RKM.conf` file must not exceed 1 MB file size.

Optionally, it might be useful to have a second IBM Security Lifecycle Manager server running for redundancy. You can duplicate your key configuration between different RKMs with the IBM Security Lifecycle Manager facilities. If so, you need to add the second RKM into your `RKM.conf` file, which adds some more lines as shown in Example 9-16.

Example 9-16 Redundant RKM configuration

```
rkmname {
...[...]
   kmipServerUri2 = tls://host:port # TLS-connection to clone number 1 to host on port
   kmipServerUri3 = tls://host:port # TLS-connection to clone number 2 to host on port
   kmipServerUri4 = tls://host:port # TLS-connection to clone number 3 to host on port
```

Note: The `RKM.conf` is provided on a per node base. Client certificates referenced in the `RKM.conf` file must be available on that node.

When you have created your `RKM.conf` file, add it into the `/var/mmfs/etc` directory. In Example 9-17, we start with a single stanza equal on all nodes so that we can distribute the RKM.conf.

Example 9-17 Distributing RKM.conf on all nodes

```
[root@ronaldo etc]# mmlsnode | tail -1 |  sed -e 's/brazil//g'
         neymar ronaldo robinho
[root@ronaldo etc]# for i in `mmlsnode | tail -1 |  sed -e 's/brazil//g'`; do scp ./RKM.conf $i:$PWD/; done
RKM.conf 100%  604     0.6KB/s   00:00
RKM.conf 100%  604     0.6KB/s   00:00
RKM.conf 100%  604     0.6KB/s   00:00
[root@ronaldo etc]#
```

The RKM.conf file is read by **mmfsd**, but only when a policy change is processed, or when the file system is mounted on that node or simply when the daemon starts.

Policy rules

By policy rule, you are configuring the encryption rules if a file gets encrypted or not. If a file is valid for encryption, this rule contains the information about the way of encryption as well. Example 9-18 provides a very simple rule to encrypt every file that matches to files containing *lala*.

Example 9-18 Simple Spectrum Scale encryption rule

```
[root@ronaldo etc]# ll
total 12
drwxr-xr-x. 2 root root 4096 Oct 17 15:39 RKMcerts
-rwxr-xr-x. 1 root root  604 Oct 17 17:54 RKM.conf
-rw-r--r--. 1 root root  247 Oct 17 17:53 rulefile
[root@ronaldo etc]# cat rulefile

rule 'thisrule' set encryption 'encr_brazil' where name like '%lala%'
rule 'thisencrule' encryption  'encr_brazil' is
  ALGO 'AES:128:CBC:FEK:HMACSHA512'
  COMBINE 'XOR'
  WRAP 'AES:ECB'
  KEYS('KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY') ### see Figure 9-13 on page 492

rule 'p1' set pool 'system' /* a default placement is required */

[root@robinho germanies_field]
```

New to the well-known policy rules is the *set encryption* phrase, which tells Spectrum Scale that this is an encryption rule. Unlike placement rules where the first match is taken by **mmfsd**, you can specify up to eight encryption rules for access patterns. The policy engine's facilities are well documented in *Spectrum Scale v4.1: Advanced Administration Guide*, SC23-7032.

The rule in Example 9-18 selects every file that corresponds to the name **lala** for encryption and applies the rule *encr_brazil* to it. If you intend to encrypt all files, put either a *%* at this point or skip the WHERE clause completely.

Within the *encr_brazil* encryption field, the daemon is shown how to encrypt:

- ALGO: The encryption algorithm.
- COMBINE: Specifies the way that MEKs are combined, in case more than one was specified in the KEYS clause.
- WRAP: The method of how the FEK is wrapped by the MEK.
- KEYS: (KEY-ID:RKM-ID)

KEY-ID is the unique identifier of a single key within the associated RKM backend. It may contain only alpha numerics, a minus sign or a dash, and can be no more than 42 characters long. You get this key from the GUI of your RKM by copying and pasting it. The RKM-ID must match the name of one of the RKM entries defined in the /var/mmfs/etc/RKM.conf on the node.

For our example, this rule means that Spectrum Scale looks in the RKM.conf for a stanza called GERMANY and out of this stanza information Spectrum Scale knows where to ask for the key (MEK). So it uses the given client certificate out of the RKM file and requests the MEK from the RKM (Figure 9-15 on page 495).

The MEK is created in step “Organizing the client’s certificate and keys in the device group” on page 491, and you can see the key as well on Figure 9-13 on page 492.

“RKM.conf” on page 495 and “Policy rules” on page 497 provide enhancements to the RKM.conf and the policy rules.

Summary and final step

If the policy is not activated by **mmfsd** (**mmchpolicy**), no files are encrypted. To activate encryption for our sample file system (germany), enable the rule to Spectrum Scale as shown in Example 9-19.

Example 9-19 Activating the encryption (placement) rule

```
[root@ronaldo etc]# mmchpolicy germany rulefile
Validated policy `rulefile': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
       0 List Rules, 0 External Pool/List Rules, 2 Encryption/Re-encryption Rules
While parsing rule 'thisencrule':
Policy `rulefile' installed and broadcast to all nodes.
[root@ronaldo etc]#
```

From now on, any new accessed file that matches **lala** is encrypted in Spectrum Scale. Existing files were not changed by this encryption rule.

9.4 Working with encryption

This section describes how to work with Spectrum Scale encryption in your cluster.

9.4.1 Verify encryption

We start by creating files on a Spectrum Scale member node in the cluster. In Example 9-20, we create some files and show how to verify if a file is encrypted. You can verify the file’s encryption status with the **mmlsattr** command.

Example 9-20 Verifying encryption

```
[root@robinho germanies_field]# pwd
/gpfs/germanies_field
[root@robinho germanies_field]# date >> file_normal
[root@robinho germanies_field]# date >> file_lala
[root@robinho germanies_field]# mmlsattr -n gpfs.Encryption file_normal
file name:            file_normal
gpfs.Encryption:      No such attribute

[root@robinho germanies_field]# mmlsattr -n gpfs.Encryption file_lala
file name:            file_lala
gpfs.Encryption:
"EAGC???u?f??????????????????(/~??????????????????)????`?R?2???+)?KEY-7ef
cd542-8559-4bdd-bfca-8d35b9f66678?GERMANY?"
EncPar 'AES:128:CBC:FEK:HMACSHA512'
        type: wrapped FEK  WrpPar 'AES:ECB'  CmbPar 'XOR'
                KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY

[root@robinho germanies_field]#
```

Note that only new generated files gets encrypted by this rule. When a node in the cluster tries to access this file, it must have a valid `RKM.conf` file and the appropriate certificate. Otherwise, a *permission denied* message is returned by Spectrum Scale.

A file that is encrypted and gets renamed, remains encrypted following the initial encryption rule from the Spectrum Scale policy. If you want to change the encryption attributes of a file, you have to use migration rules.

Add nodes to IBM Spectrum Scale cluster with encrypted file systems

If you need to add nodes to the cluster, they need to have the GSkit and crypto-Spectrum Scale packages installed, the appropriate RKM.config, and the client certificates to access the encrypted files. However, nodes that do not have crypto packages installed can **`mmchpolicy`** to a file system whether or not it is encrypted, and can **`mmlspolicy`** to the file system even though they cannot mount it.

In addition, a node without crypto packages (and hence unable to mount the file system) can run offline **`mmfsck`** on an encrypted file system.

9.4.2 Back up your keys

If the MEKs stored in the RKM are lost, the encrypted files in the Spectrum Scale file systems are no longer able to be decrypted. The data will be lost. Consequently, you should take a backup before and after any operation on the IBM Security Key Lifecycle Manager server once you have configured encryption with Spectrum Scale, and save the backup file in a separate and secure location. If an attacker obtains access to the key server or to the backup of the key server, the attacker will be able to decrypt your data.

To create a backup, connect to the WebGUI, select the backup register, and follow the instructions as shown in Figure 9-16.

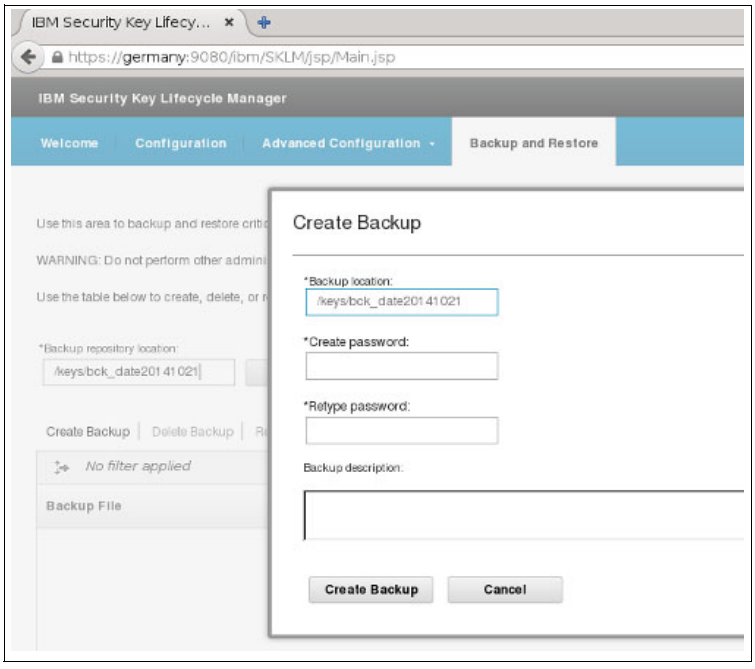


Figure 9-16 Backing up the key/certificate configuration

9.4.3 Remote key manager and RKM.conf

The `RKM.conf` file is read every time a file system is mounted. If no changes are made to the Spectrum Scale policy engine or any other rules are applied to Spectrum Scale, the RKM file is not read again while Spectrum Scale is running. There are more reasons that force Spectrum Scale to read the RKM file, for example, an outage of a node which enforces several recovery actions including remounting file systems. Therefore, it is good to have a backup RKM in place. The IBM Security Key Lifecycle Manager supports replication of the configuration (keys) to a remote RKM so that you can build disaster recovery capable environments. For more information, refer to the following publications from:

- `http://www.ibm.com/software/tivoli/library`
- IBM Security Key Lifecycle Manager Quick Start Guide, GI13-2316
- IBM Security Key Lifecycle Manager Installation and Configuration Guide, SC27-5335

If the connection to your RKM is broken and you try to start your cluster or mount an encrypted file system, you might see error messages similar to the one in Example 9-21. This might be the reason for data in your file system to be unavailable.

Example 9-21 Typical error messages, when RKM is unavailable

```
Tue Oct 21 15:40:31.569 2014: [W] The key server germany (port 5696) had a failure and will be quarantined
for 1 minute(s).
Tue Oct 21 15:40:31.570 2014: [E] Unable to open encrypted file: inode 40973, fileset 1, file system
germany.
Tue Oct 21 15:40:31.571 2014: [E] Key 'KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY' could not be
fetched. The TCP connection with the RKM could not be established.
```

In this case, Spectrum Scale tries periodically to get the keys from RKM and once the issues on RKM or the network are solved, you can access your files automatically.

9.4.4 Secure deletion file data

Now you can use Spectrum Scale encryption capabilities for achieving secure deletion by simply destroying the appropriate MEKs.

A common use case might come from various security compliance guidelines, where data on disk needs to destroyed several times. There are some disk technologies (Flash/SSD) that are unable to guarantee that all sectors had been overwritten, but now you can use Spectrum Scale encryption to address secure deletion.

To demonstrate how secure deletion works, we create four files in our example. The FEK of all of these files is encrypted with the same MEK. We describe now how to assure the secure deletion of just one out of these four files. Let us assume file4 should be deleted while all the other files should remain accessible (files are encrypted), you would have to:

1. Create a new KEY on the RKM (already done) as shown in Figure 9-13 on page 492.
2. Change the policy so that for all new created files, its FEK is wrapped with the new MEK.
3. Delete the files and data with the POSIX command **`rm`** or with any other tools you prefer.
4. Provide a rule for rewrapping the FEK of all the files having the old MEK used.
5. Apply the rule.
6. Delete the old key from RKM when the rewrapping is finished.
7. Force the GPFS daemons to evict the old MEK out of the cache.

By following the preceding procedure, it is assured that even if some data, meaning some segments on disk, are physically present, nobody could ever read this data because the MEK is unavailable.

To explain and demonstrate it in Example 9-22, the file is not deleted by the UNIX **rm** command to show that it cannot be read any more, since the old MEK was removed.

Example 9-22 shows how a secure deletion can be achieved. Starting with the focus on two files, which are matching to the existing file system encryption rule. You can see the file `lala` and the file `lala_todelete` having the appropriate encryption attributes. To have more clear names, the file `lala_todelete` is removed and as you can see, the encryption attributes are kept because they belong to the file's inode and the EA will not change by changing a file's name.

Example 9-22 How a secure deletion can be achieved

```
[root@ronaldo germanies_field]# mmlsattr -n gpfs.Encryption lala_todelete
lalalala
file name:            lala_todelete
gpfs.Encryption:[...] KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY
file name:            lalalala
gpfs.Encryption:[...] KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY

[root@ronaldo germanies_field]# mv lala_todelete regular_name
[root@ronaldo germanies_field]# mmlsattr -n gpfs.Encryption regular_name
file name:            regular_name
gpfs.Encryption:[...] KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY

[root@ronaldo germanies_field]#
```

If not done, create a new key on the RKM GUI in the device group that your node belongs to. To be more accurate, create a new MEK in the device group you selected with your `RKM.conf` file and policy rule.

We already had created some more keys done in "Organizing the client's certificate and keys in the device group" on page 491. You should see similar output in your GUI in comparison to Figure 9-17 (your active MEK from your configuration and at least one additional key).

Figure 9-17 Available MEKs

The next step is to change the default rule in the file system such that every new incoming file gets the new encryption MEK applied. We now take the second key from Figure 9-17 and change the policy rules of Spectrum Scale as shown in Example 9-23.

Example 9-23 Update the policy rule file with a new MEK

```
[root@ronaldo etc]# cat rulefile
rule 'thisrule' set encryption 'encr_brazil' where name like '%lala%'
```

```
rule 'thisencrule' encryption  'encr_brazil' is
  ALGO 'AES:128:CBC:FEK:HMACSHA512'
  COMBINE 'XOR'
  WRAP 'AES:ECB'
  /** KEYS('KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY')   old key */
  KEYS('KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY')

rule 'p1' set pool 'system' /* this is currently required */

[root@ronaldo etc]#
```

After activating the policy using **mmchpolicy** to Spectrum Scale, verify that the new files are written with the new key. You can check by using the **mmlsattr** command as shown in Example 9-24.

Example 9-24 mmchpolicy and mmlsattr commands

```
[root@ronaldo etc]# mmchpolicy germany rulefile
Validated policy `rulefile': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules, 2 Encryption/Re-encryption Rules
Policy `rulefile' installed and broadcast to all nodes.
[root@ronaldo etc]#

[root@ronaldo germanies_field]# echo "haaaaaaloo" >> lala_new
[root@ronaldo germanies_field]# mmlsattr -n gpfs.Encryption lala_new
file name:            lala_new
gpfs.Encryption:[...] KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY
```

Now you need to rewrap the FEK of all the already existing files in the file system to use this new key so that later on, the old key can be deleted. Usually, you would take the same access pattern in the rewrap rule for selecting the files just like you did before when creating the files. Example 9-25 shows how to rewrap every file match to **lala**. Keep in mind that we renamed our file for delete to *regular_name*, which does not apply to this rule to show that secure deletion works. In real scenarios, you probably would have already deleted the file.

Example 9-25 Key rewrap rule

```
[root@ronaldo etc]# cat rewrap_keys
RULE 'rewrap' CHANGE ENCRYPTION KEYS FROM
'KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY'
                 to 'KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY'
WHERE name like '%lala%'
```

By applying the rule-file to Spectrum Scale, you force that every file matching to the rule gets its FEK rewrapped with the new key (MEK). This policy scan *does not* need to read the data of all files and re-encrypt the data. The scan only affects the extended attributes of the files, rewrapping the FEK with the new MEK to replace the old one. Although it is still expensive to scan a large file system, it is a lot cheaper just to scan and change the metadata information than reading and rewriting every file's data.

Use the **mmapplypolicy** command to rewrap the keys as shown in Example 9-26.

Example 9-26 Using the mmapplypolicy command

```
[root@ronaldo etc]# mmapplypolicy germany -P rewrap_keys
[...]
[I] Loaded policy rules from rewrap_keys.
[...] some output repressed for better overview [...]
```

```
I] 2014-10-20@20:49:05.229 Policy evaluation. 10 files scanned.
[I] 2014-10-20@20:49:05.232 Sorting 7 candidate file list records.
[I] 2014-10-20@20:49:05.233 Choosing candidate files. 7 records scanned.
[I] Summary of Rule Applicability and File Choices:
 Rule#      Hit_Cnt          KB_Hit          Chosen       KB_Chosen          KB_Ill     Rule
     0            7             224               7             224               0     RULE 'rewrap' CHANGE
ENCRYPTION KEYS FROM KEY 'KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY' TO KEY
'KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY' WHERE(.)
[...]

Pool_Name                   KB_Occupied        KB_Total  Percent_Occupied
system                          2665472        41943040      6.354980469%
[I] 2014-10-20@20:49:05.964 Policy execution. 7 files dispatched.
[I] A total of 7 files have been migrated, deleted or processed by an EXTERNAL EXEC/script;
       0 'skipped' files and/or errors.
[root@ronaldo etc]#

[root@ronaldo germanies_field]# mmlsattr -n gpfs.Encryption lala_new lala_file3 regular_name
file name:            lala_new
gpfs.Encryption:[..]  KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY
file name:            lala_file3
gpfs.Encryption:[...] KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY
file name:            regular_name
gpfs.Encryption:[...] KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY

[root@ronaldo germanies_field]#
```

As you can see in Example 9-26 on page 502, the keys of **lala** matching files had been changed now to the new MEK, while the `regular_name` file still has the old key. This is now the point where we remove the key from RKM to make sure that nobody can read the data that was encrypted with the old key.

For this, enter the GUI in the RKM, select the appropriate device group, and right-click. You should get a fast access menu for “Manage keys and devices”. Select the old key on the right side of the window and remove the key as shown in Figure 9-18.

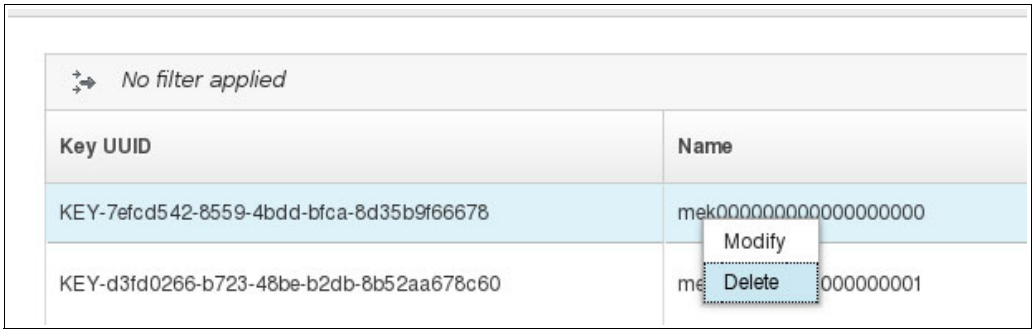


Figure 9-18 Removing the key from RKM

This is a very good example to demonstrate that Spectrum Scale still does know the old MEK because it is held in cache. If you look in Example 9-27 on page 504, you see that the file can still be read. Therefore, we do have to command Spectrum Scale to invalidate the old key in the cache. Do not get confused because even after purging the old MEK from cache, the content of the file is accessible. Usually, you would not see the file at all because it is already deleted. In our example, the file itself is still cached in the pagepool and data in the pagepool

is not encrypted at all. So we would have to clear the file buffer cache. But as stated, it is just an example and we did not delete the file to demonstrate how secure deletion works.

Note: You need to purge old MEKs from the GPFS daemon.

Example 9-27 Purging the old MEK from the GPFS daemon cache

```
[root@ronaldo germanies_field]# cat regular_name
Mon Oct 20 16:10:23 EDT 2014

[root@ronaldo germanies_field]# /usr/lpp/mmfs/bin/tsctl encKeyCachePurge
'KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY'
Key KEY-7efcd542-8559-4bdd-bfca-8d35b9f66678:GERMANY purged from the cache.
[root@ronaldo germanies_field]# cat regular_name
Mon Oct 20 16:10:23 EDT 2014    <<<<<<<<< file buffer from pagepool,
```

To finally finish your secure deletion example, clear the daemon cache on the node by remounting the file system. Then, it is sure that the file is not in cached. So you see that the file cannot be read from disk any more because the old MEK is not available inside the mmfsd and not available on RKM as well. However, the file can be deleted as the inode is never encrypted and the access is controlled by regular file access permissions: mode bits and ACLs. Refer to Example 9-28.

Example 9-28 Clearing the cache and verifying that the file is not readable

```
root@ronaldo ~]# mmumount germany
Mon Oct 20 16:59:17 EDT 2014: mmumount: Unmounting file systems ...
[root@ronaldo ~]# mmmount germany
Mon Oct 20 16:59:22 EDT 2014: mmmount: Mounting file systems ...
[root@ronaldo ~]# cd -
/gpfs/germanies_field
[root@ronaldo germanies_field]# cat regular_name
cat: regular_name: Operation not permitted
[root@ronaldo germanies_field]#

[root@ronaldo germanies_field]# rm regular_name
rm: remove regular file `regular_name'? y
[root@ronaldo germanies_field]#
```

9.4.5 Understanding read access to an encrypted file

Before a file can be read, it has to be opened first. During the file opening, the MEK is taken to unwrap the FEK. The file can be accessed if the FEK can be obtained (see Figure 9-19 on page 505 for more detailed information). The FEK source must be unwrapped at least and at most once. More than one attempt might be required depending on the RKM credentials deployed on the node attempting the operation.

FEK1 and FEK2 are cached in the in-memory file descriptor and remain in memory until the file is closed.

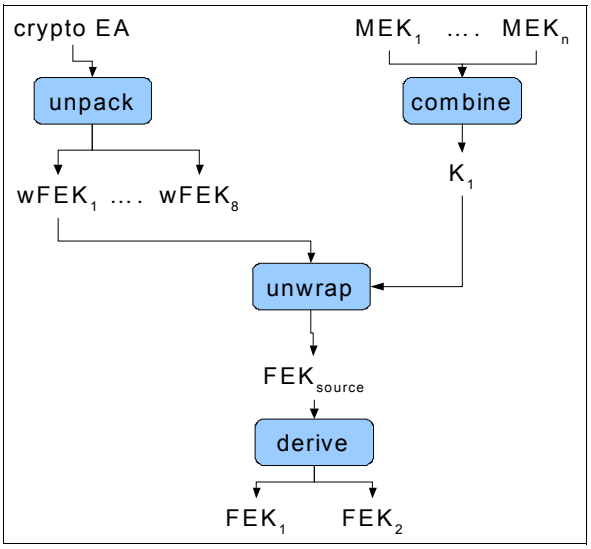


Figure 9-19 Read access to an encrypted file

When the file is opened, the FEK is cached and is taken to encrypt every single 512-Byte sector before it is written to the pagepool or handed over to the upper layers.

9.5 Implementing access control on a node base

As stated before, you can apply up to eight different rules to a file in the policy file. However, keep in mind that your rule file must not exceed 1 MB overall size. All encryption rules are applied which means the FEK of the file is wrapped with all of the applied MEKs. This gives us the possibility to build up cluster up to eight nodes, where every node in the cluster owns just its unique certificate and no common client certificate is allowed.

In larger clusters with more than eight nodes, you can create several node groups that have access to certain subsets of data in the file system, maybe bounded on a file system or any other access pattern. To demonstrate this in our example, we now enhance our setup by using additional client certificates. This means that we establish encryption rules that apply to a certain subset of files and these files are accessible by every node, and we demonstrate that we can generate access control by node, and how to prevent access to a file from a node. Refer to Figure 9-20 on page 506 again as a good starting point for overview.

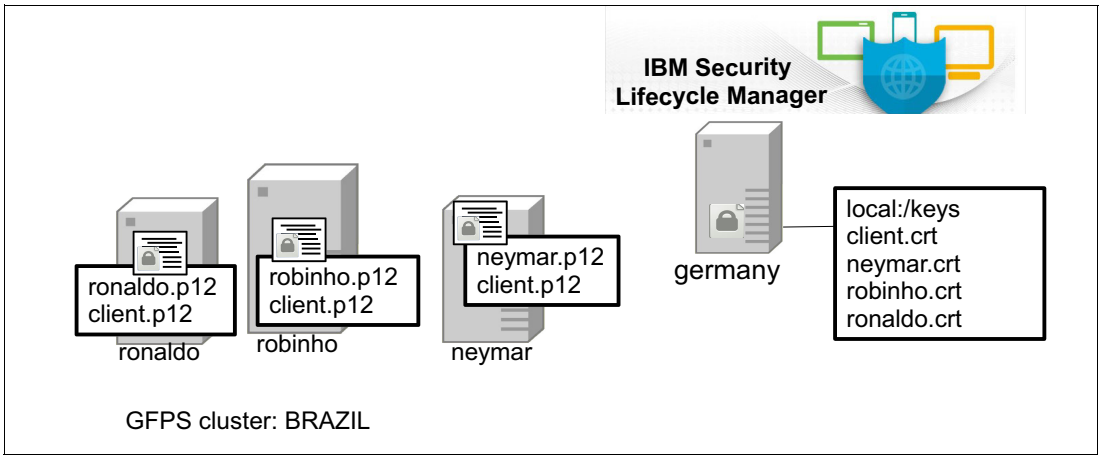


Figure 9-20 Establishing new encryption rules among the clients

9.5.1 Changes to RKM

We enhance our setup now with a new device group inside the RKM. We create another Spectrum Scale device group via the WebGUI, and give it the name GPFS_BRAZIL_HALF. After that, start again from the welcome window, select your new device group, and by right-clicking, you should get a fast access menu for *Manage keys and devices*. Create one or a few new keys and put your unique client certificates in that DG as well. In our example, we put in the client's certificate from Ronaldo and Robinho and we do not put in the certificate from Neymar. Finally, your new device group on the RKM should look similar to Figure 9-21.

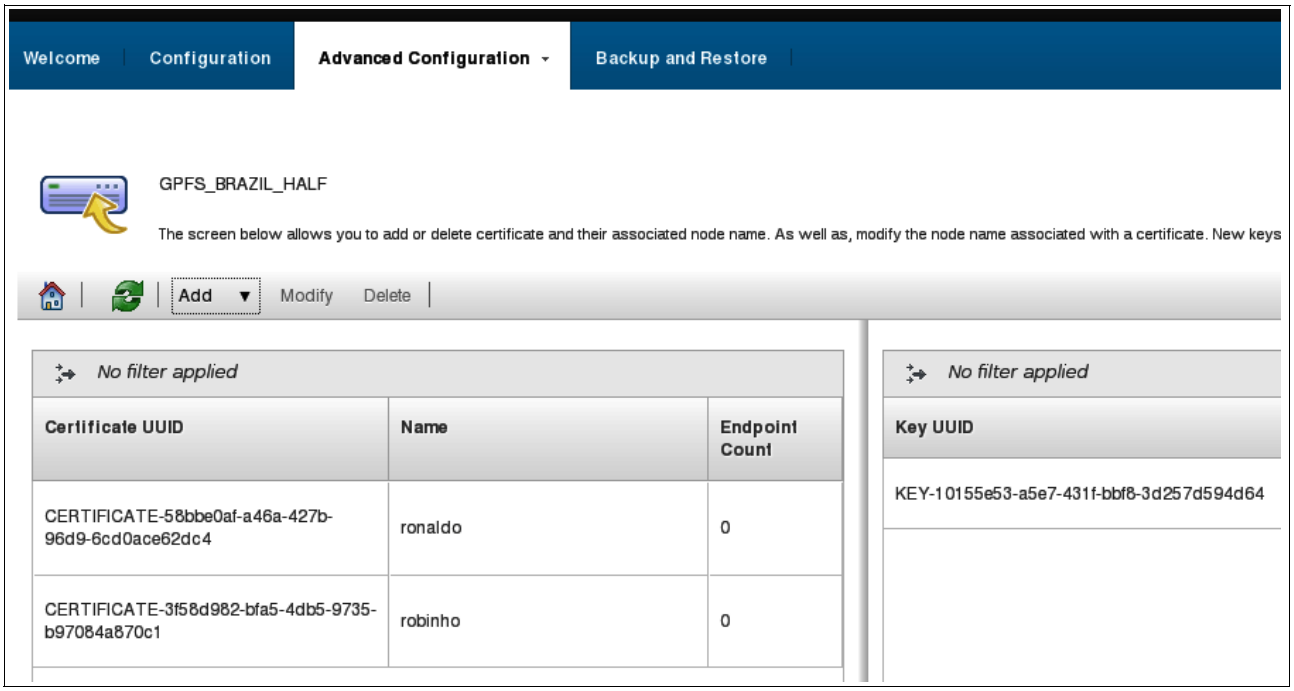


Figure 9-21 WEBGUI screen capture from GPFS_BRAZIL_HALF

To explain what it means: Only Nodes with the certificates of *robinho* or *ronaldo* have access to this key (right side of the picture) in the device group BRAZIL_HALF.

Furthermore, we add NEYMARs certificate to the DG BRAZIL to achieve the following situation regarding the key/certificate access control (Figure 9-22).

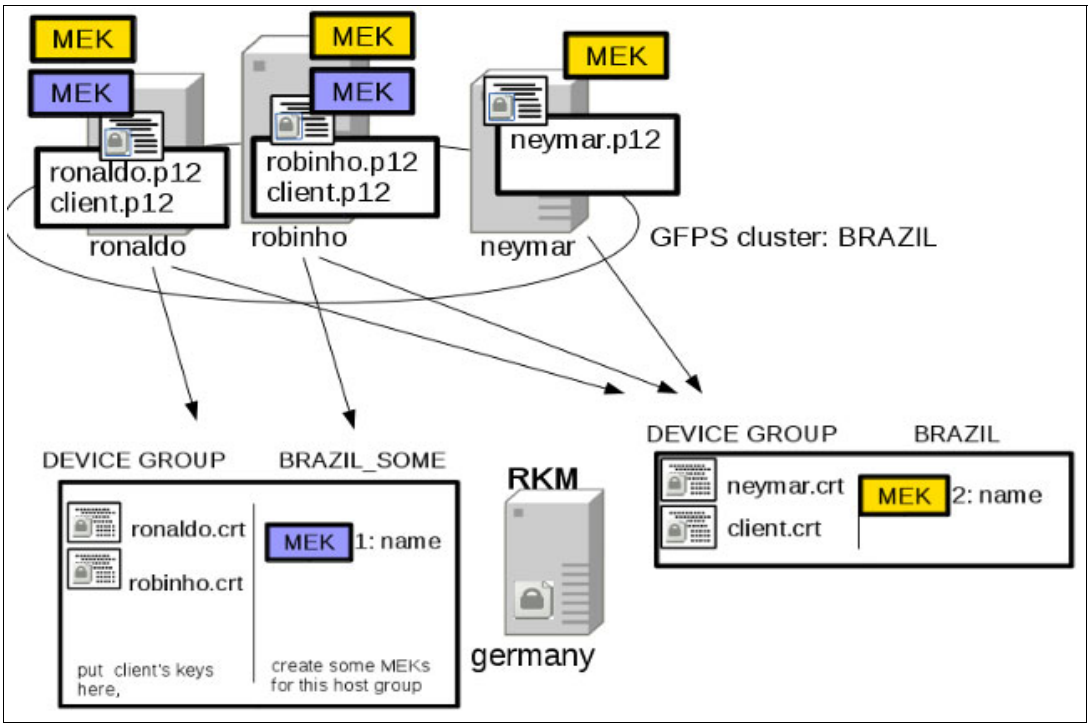


Figure 9-22 Organize node certificates

Note: A certificate can belong to just one DG only. If you need to connect a node in more than one DG in parallel, create more certificates for that node.

9.5.2 Changes to RKM.conf

Next step is to change the Spectrum Scale configuration accordingly. We have to enhance the existing policy rule. So we add a rule that all files placed in a certain fileset of the file system get encrypted by the MEK out of this new second device group. Therefore, an update to the `RKM.conf` file has to follow as well. As a good starting point, we add the new key information to the new `RKM.conf` file first, and then make the appropriate changes to the file system policy rules.

Note: To apply encryption policy rule sets to Spectrum Scale, the `RKM.conf` file is read. Therefore, all client certificates referenced in the local available `RKM.conf` of a node must be available on that node.

A rule referencing to a non-existing MEK by a missing entry in the RKM.conf generates warnings but might be OK as intended.

The `RKM.conf` is read only in the moment when the **`mmchpolicy`** command is running and activates the policy to Spectrum Scale.

Every node gets its own unique `RKM.conf` file according to the available client keys. In our example, ROBINHO, and RONALDO get new `RKM.conf` files by adding a new stanza

NEWACCESS. NEYMAR gets its own unique enhancement according to its certificates as shown in Example 9-29.

From now on, every node has to have its own `RKM.conf` file because its entries must correspond to the local available client certificates.

Example 9-29 Enhance the RKM.conf files

```
root@ronaldo etc]# cat RKM.conf
GERMANY {
  type = ISKLM                                  # RKM is a ISKLM server
   kmipServerUri = tls://germany:5696           # TLS-connection to host on port
   keyStore = /var/mmfs/etc/RKMcerts/client.p12 # The path to the PKCS12 file containing server certificat
   passphrase = beer                            # Passphrase of private key
   clientCertLabel = client                     # Label of the client keypair to be used among those in the
                                                # keystore
   tenantName = GPFS_BRAZIL                     # Name of tenant, set in ISKLM set-up
}

NEWACCESS {
 type = ISKLM
   kmipServerUri = tls://germany:5696
   keyStore = /var/mmfs/etc/RKMcerts/ronaldo.p12
   passphrase = beer
   clientCertLabel = ronaldo
   tenantName = GPFS_BRAZIL_HALF
}

# and on Robinho, we keep the GERMANY section and add...
[root@robinho etc]# cat RKM.conf
GERMANY {
[...] output repressed for better overview [...] }

NEWACCESS{
 type = ISKLM
   kmipServerUri = tls://germany:5696
   keyStore = /var/mmfs/etc/RKMcerts/robinho.p12
   passphrase = beer
   clientCertLabel = robinho
   tenantName = GPFS_BRAZIL_HALF
}

# and on neymar we keep the GERMANY section and add a dedicated section called
#NEYMAR:
[root@neymar etc]# cat RKM.conf
GERMANY {
[...] output repressed for better overview [...] }

NEYMAR {
  type = ISKLM                                  # RKM is a ISKLM server
   kmipServerUri = tls://germany:5696           # TLS-connection to host on port
   keyStore = /var/mmfs/etc/RKMcerts/neymar.p12 # The path to the PKCS12 file containing server certificat
   passphrase = beer                            # Passphrase of private key
   clientCertLabel = neymar                     # Label of the client keypair to be used among those in the
                                                # keystore
   tenantName = GPFS_BRAZIL                  # Name of tenant, set in ISKLM set-up
}
```

For better demonstration purposes, we create a fileset. There is no need to have a fileset for the encryption. It is just another example to demonstrate the various flexible access patterns of the policy engine. In addition, it might be useful to bound certain application workloads to dedicated filesets.

9.5.3 Changes to Spectrum Scale policy

Now we are ready for updating the policy rule file. In Example 9-30, we change such that every file gets encrypted with the common MEK, and accessible by the common (shared) client certificate. In addition, we enhance the rule that files going to a certain fileset are wrapped again with one further key. We show how to configure exceptions for the encryption rules.

We strongly advise making a copy of the active policy rules with the **mmlspolicy** command before starting to add any new rules. Then, add a section to meet your needs for the files with limited access as shown in Example 9-30.

Note: If you intend to use the documentation for copy and paste, check that your character coding is set appropriately to interpret the quote as a real single quotation mark >> ‘ <<.

Example 9-30 Changing the policy rule

```
[root@ronaldo etc]# cat rulefile
rule 'thisrule' set encryption 'encr_brazil' where name like '%'
rule 'thisencrule' encryption  'encr_brazil' is
  ALGO 'AES:128:CBC:FEK:HMACSHA512'
  COMBINE 'XOR'
  WRAP 'AES:ECB'
  KEYS('KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY')

RULE 'Do not encrypt files with extension goal'
SET ENCRYPTION EXCLUDE
FOR FILESET('halftime')
WHERE NAME LIKE '%.goal'

rule 'thisrule3' set encryption 'encr_brazil3' FOR FILESET('halftime')
rule 'next3' encryption  'encr_brazil3' is
  ALGO 'AES:128:CBC:FEK:HMACSHA512'
  COMBINE 'XOR'
  WRAP 'AES:ECB'
  KEYS('KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS')

rule 'p1' set pool 'system' /* this is currently required */
```

As you can see in the rule, we reference to a dedicated fileset that we created meanwhile. When activating the rule, we get some warnings. This is correct because not all nodes have the key that the rule is referencing to (Example 9-31 on page 510). The warnings should match your Node configuration. In our example, NEYMAR does not have the KEY with the UID *KEY-10155e53-a5e7-431f-bbf8-3d257d594d64* because NEYMAR has no client certificate nor a stanza in the RKM.conf to get this key. Regardless of which node within the cluster initiates the **mmchpolicy**, the warning messages that you get correspond to the node who is missing the key, which is the case for one node, NEYMAR.

Example 9-31 Activating the Spectrum Scale policy rule file

```
[root@ronaldo etc]# mmchpolicy germany rulefile
Validated policy `rulefile': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules, 5 Encryption/Re-encryption Rules
While parsing rule 'next3':
[W] The RKM identifier specified for key 'KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS' is
nonexistent; check the RKM.conf file.
Policy `rulefile' installed and broadcast to all nodes.
[root@ronaldo etc]#
```

9.5.4 Activating access control

With the **mmchpolicy** command, the new rule is automatically enabled and valid for all new generated files. With the following examples, we complete the demonstration and give some additional details to show how it works (Example 9-32). Because of the first rule that every file should be encrypted with the common known MEK, which is related to client.p12 and the GERMANY RKM.conf stanza, every node should still be able to access the files.

Example 9-32 Verifying access control

```
root@ronaldo etc]# mmlsfileset germany
Filesets in file system 'germany':
Name                    Status    Path
root                    Linked    /gpfs/germanies_field
halftime                Linked    /gpfs/germanies_field/halftime
[root@ronaldo etc]# cd /gpfs/germanies_field/halftime
[root@ronaldo halftime]# date > lets
[root@ronaldo halftime]# date > play
[root@ronaldo halftime]# date > soccer
[root@ronaldo halftime]# ls -ltr
total 32
-rw-r--r--. 1 root root 29 Oct 21 14:09 lets
-rw-r--r--. 1 root root 29 Oct 21 14:09 play
-rw-r--r--. 1 root root 29 Oct 21 14:09 soccer
[root@ronaldo halftime]# ssh robinho "cat $PWD/lets"
Tue Oct 21 14:09:02 EDT 2014
[root@ronaldo halftime]#
```

As you can see, all nodes are able to access the file. By demonstrating that even NEYMAR can access the file, you understand that it is enough to have just one set of MEKs (in this case the FEK is wrapped with one MEK at a time only). See Example 9-33 to verify that an FEK can be wrapped with different MEKs.

Example 9-33 One file, one FEK, but wrapped several times

```
[root@ronaldo halftime]# ssh neymar "cat $PWD/lets"
Tue Oct 21 14:09:02 EDT 2014
[root@ronaldo halftime]# mmlsattr -n gpfs.Encryption lets
file name:            lets
gpfs.Encryption:      "EAGC??????????????????????????r>-?
z'????????????????`S???~????W:!?)?KEY-d3fd0266-b723-48be-b2db-8b52aa678c60?GERMANY??????????????QZ
sO?z?n??m@??)?KEY-10155e53-a5e7-431f-bbf8-3d257d594d64?NEWACCESS?"
EncPar 'AES:128:CBC:FEK:HMACSHA512'
        type: wrapped FEK  WrpPar 'AES:ECB'  CmbPar 'XOR'
                KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY
        type: wrapped FEK  WrpPar 'AES:ECB'  CmbPar 'XOR'
```

```
                KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS

[root@ronaldo halftime]#
```

So depending on the placement according to the Spectrum Scale policy rule and the key configuration on the RKM, Spectrum Scale controls whether a node can access a file or not.

Now we want to eliminate access from a node, so we change the default rule in the policy that applies to a limited search pattern only. While the file resides in the fileset, all files still get encrypted. Change the rule (only the first line) and apply them all as shown in Example 9-34.

Example 9-34 Changing the rule file

```
root@ronaldo etc]# cat rulefile # change only the first line
rule 'only some files' set encryption 'encr_brazil' where name like '%some%'
[...] some out put repressed, [...]
[...] rest is the same like in Example 9-30 on page 509 [...]

[root@ronaldo etc]# mmchpolicy germany rulefile
Validated policy `rulefile': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules, 5 Encryption/Re-encryption Rules
While parsing rule 'next3':
[W] The RKM identifier specified for key 'KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS' is
nonexistent; check the RKM.conf file.
Policy `rulefile' installed and broadcast to all nodes.
[root@ronaldo etc]#
```

By applying this rule, new generated files do not get the MEK out of the GERMANY stanza in the `RKM.conf` file any longer. If the files are generated in the fileset, they will be encrypted by the other MEK (NEWACCESS). On RONALDO, we step into the fileset's path and create some files. By opening an SSH to ROBINHO, you can prove that the other node is able to access the file as well. Now you can see, while ROBINHO still can access the file, NEYMAR cannot. Because the file is in the given fileset and the necessary client key and RKM.conf are not available on node NEYMAR, an *Operation not permitted* message is shown as presented in Example 9-35.

Example 9-35 Accessing the files if the operation is permitted

```
[root@ronaldo halftime]# date > try_to_win
[root@ronaldo halftime]# ls -lrt
total 96
-rw-r--r--. 1 root root 29 Oct 21 14:09 lets
-rw-r--r--. 1 root root 29 Oct 21 14:09 play
-rw-r--r--. 1 root root 29 Oct 21 14:09 soccer
-rw-r--r--. 1 root root 29 Oct 21 14:25 try_to_win
[root@ronaldo halftime]# mmlsattr -n gpfs.Encryption try_to_win
file name:            try_to_win
gpfs.Encryption:[...] KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS


[root@ronaldo halftime]# ssh robinho "cat $PWD/try_to_win"
Tue Oct 21 14:25:37 EDT 2014
[root@ronaldo halftime]# ssh neymar "cat $PWD/try_to_win"
cat: /gpfs/germanies_field/halftime/try_to_win: Operation not permitted
[root@ronaldo halftime]#
```

Considering the exception rule that we created, every file within this fileset that corresponds to a name ending to ~.goal should be skipped by the encryption rule next3, and it is encrypted

by the default rule only. This means in our demonstration here that the files are not encrypted because they do not match to *~some~* and so every file ending up to goal is accessible by every node. See Example 9-34 on page 511 again to understand the policy rules.

We show the EXCLUDE clause in Example 9-36, and you can see that the files are not encrypted, thus NEYMAR can see at least these goals files.

Example 9-36 Exclude files from encryption

```
[root@ronaldo halftime]# for i in 1 2 3 4 5 6 7
> do
> echo "HURAAAAAA JIPIIII" > $i.goal
> done
[root@ronaldo halftime]# ll
total 128
-rw-r--r--. 1 root root 18 Oct 21 14:46 1.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 2.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 3.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 4.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 5.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 6.goal
-rw-r--r--. 1 root root 18 Oct 21 14:46 7.goal
-rw-r--r--. 1 root root 29 Oct 21 14:09 lets
-rw-r--r--. 1 root root 29 Oct 21 14:09 play
-rw-r--r--. 1 root root 29 Oct 21 14:09 soccer
-rw-r--r--. 1 root root 29 Oct 21 14:25 try_to_win
[root@ronaldo halftime]# ssh ROBINHO "cat $PWD/2.goal"
HURAAAAAA JIPIIII
[root@ronaldo
 halftime]# ssh NEYMAR "cat $PWD/2.goal"
HURAAAAAA JIPIIII
[root@ronaldo halftime]# mmlsattr -n gpfs.Encryption 1.goal
file name:            1.goal
gpfs.Encryption:      No such attribute

[root@ronaldo halftime]#
```

9.5.5 Rewrap FEKs to achieve node isolation from data

Rewrapping policies are used to change the way a set of FEKs are encrypted. That is, to change the set of MEKs that wrap the FEKs of those files. Rewrapping applies only to files that are already encrypted, and the rewrapping operation acts only on the gpfs.Encryption EA of the files. Rewrapping is done by using the **mmapplypolicy** command to apply a set of policy rules containing one or more CHANGE ENCRYPTION.

You can use rewrapping for assigning new keys to a subset of existing encrypted files. By this, you can change and limit the access to the files to a subset of nodes having the appropriate RKM.conf and client certificate available.

You can apply rewrapping rules without changing the default policy of the Spectrum Scale file system. So we continue our example to demonstrate the encryption capabilities in terms of fine granular control mechanism, and how to prevent access to files from certain nodes.

Note: The rewrapping nodes (specified with **mmapplypolicy -N**) must have access to both the old key and the new key.

Assume that after a while it might be better to prevent the access to a certain file in the fileset. Then, we simply change the MEK of that single file by a policy and rewrap these files with a new key. This access control mechanism is nearly similar to the example from 9.5, “Implementing access control on a node base” on page 505, but this time we do not change the Spectrum Scale rule, we just apply one rewrap migration rule by the node NEYMAR. For a simple rewrap rule, see Example 9-37, which shows that NEYMWAR can access the file because he has the MEK (UID *KEY-d3fd0266-b723-48be-b2db-8b52aa678c60*). Thus, we apply the rewrap rule as shown in Example 9-37.

Example 9-37 Creating a file and applying a rewrap rule

```
[root@ronaldo halftime]# ssh neymar "cat $PWD/against_one.goal"
 ok ok
[root@ronaldo halftime]# mmlsattr -n gpfs.Encryption against_one.goal
file name:            against_one.goal
gpfs.Encryption:[...] KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY

[root@ronaldo halftime]# cat /var/mmfs/etc/rewrap_keys.small
RULE 'rewrap' CHANGE ENCRYPTION KEYS FROM 'KEY-d3fd0266-b723-48be-b2db-8b52aa678c60:GERMANY'
               to 'KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS'
WHERE name like '%against_one.goal%'
[root@ronaldo halftime]# mmapplypolicy germany -P /var/mmfs/etc/rewrap_keys.small
 some out repressed for better overview...
```

Now look at the key from the file `against_one.goal` as shown in Example 9-38. It has the other MEK now, and therefore, NEYMAR no longer can access the file.

Example 9-38 Key from the file against_one.goal

```
[root@ronaldo halftime]# mmlsattr -n gpfs.Encryption against_one.goal
file name:            against_one.goal
gpfs.Encryption:[...] KEY-10155e53-a5e7-431f-bbf8-3d257d594d64:NEWACCESS

[root@ronaldo halftime]# ssh neymar "cat $PWD/against_one.goal"
cat: /gpfs/germanies_field/halftime/against_one.goal: Operation not permitted
[root@ronaldo halftime]#
```

9.5.6 Securing your environment

Consider your system security settings as well. If you are providing access control by encryption, secure the access within your inter-node communication as well. If a node would get access to a foreign RKM.conf or foreign client certificates, the node would be able to get access to the appropriate files as well by changing the configuration locally.

9.5.7 Summary

There are much more various configuration scenarios that are possible. You can take this example to see how encryption works on different levels, so that even though every file is encrypted at all, only a subset of nodes can access the data within the cluster, depending on the RKM configuration.

As you can read in this chapter, the encryption configuration can become very complex easily. The following main facts provide a summary:

- A client certificate can belong to one DG in the RKM only.

- Client certificates can be equal among any or several nodes.
- FEK can be wrapped up to eight times with eight MEKs.
- Files can be accessed as long one valid MEK is available on the node doing the access.
- All KEYs referenced by a rule are resolved by the `RKM.conf` file.
- The `RKM.conf` file does not need to be identical on the nodes.
- References to a key in the `RKM.conf` file imply to have the appropriate certificate available.

The examples shown in this chapter provide an overview of how encryption works within Spectrum Scale. However, deeper background information about policy rules and further encryption algorithm implemented in Spectrum Scale is available on the official product documentation website:

`http://www-01.ibm.com/support/knowledgecenter/SSFKCN/gpfs_welcome.html`

A

Recovery of the IBM Spectrum Scale file system configuration

This appendix provides extended examples and scripts for restoring the IBM Spectrum Scale file system configuration from **`mmbackupconfig`** information.

The following topics are discussed in this section:

- Back up the file system configuration
- Generating the scripts for creating the original file system configuration
- Restoring the file system configuration

© Copyright IBM Corp. 2015. All rights reserved.

Back up the file system configuration

The backup of the file system configuration is performed by the **mmbackupconfig** command. Example A-1 uses a Spectrum Scale file system named gpfs1 and shows the output command execution.

Example A-1 Running mmbackupconfig for gpfs1 file system

```
root@fsaix4:/>mmbackupconfig gpfs1 -o /tmp/gpfs1.backup

mmbackupconfig: Processing file system gpfs1 ...
mmbackupconfig: Command successfully completed
```

The configuration data is saved in an ASCII file with a specific format. See the content of the file in Example A-2.

Example A-2 Content of the mmbackupconfig output file

```
root@fsaix4:/>cat /tmp/gpfs1.backup
%%8888%%:00_BK_HEADER:1103:gpfs1:Tue Oct 28 20%3A52%3A44 EDT 2014::tsmgpfs.tsmsrv:::::::::
%%9999%%:00_VERSION_LINE::1410:3:78::lc:tsmsrv::5:/usr/bin/ssh:/usr/bin/scp:12742445374757761618:lc2:1413575263::tsmg
pfs.tsmsrv:2:1:1:2:A:::central:0.0:
%%home%%:10_NODESET_HDR:::4:TCP:AUTHONLY:1191:::13:1410:1410:A:4:::6:6:6:3:::::::
%%home%%:70_MMFSCFG::1:clusterName:tsmgpfs.tsmsrv:::::::::::::::::::::
%%home%%:70_MMFSCFG::2:clusterId:12742445374757761618:::::::::::::::::::::
%%home%%:70_MMFSCFG::3:autoload:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::4:dmapiFileHandleSize:32:::::::::::::::::::::
%%home%%:70_MMFSCFG::5:minReleaseLevel:1410 4.1.0.4:::::::::::::::::::::
%%home%%:70_MMFSCFG::6:ccrEnabled:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::7:tiebreakerDisks:tsmdisk1:::::::::::::::::::::
%%home%%:70_MMFSCFG::8:prefetchThreads:100:::::::::::::::::::::
%%home%%:70_MMFSCFG::9:pagepool:1500M:::::::::::::::::::::
%%home%%:70_MMFSCFG::10:cipherList:AUTHONLY:::::::::::::::::::::
%%home%%:70_MMFSCFG::11:clusterManagerSelection:preferManager:::::::::::::::::::::
%%home%%:30_SG_HEADR:gpfs1::151:no:::0:::no:::::::::::::::
%%home%%:40_SG_ETCFS:gpfs1:1:%2Fgpfs1:
%%home%%:40_SG_ETCFS:gpfs1:2:   dev             = /dev/gpfs1
%%home%%:40_SG_ETCFS:gpfs1:3:   vfs             = mmfs
%%home%%:40_SG_ETCFS:gpfs1:4:   nodename        = -
%%home%%:40_SG_ETCFS:gpfs1:5:   mount           = mmfs
%%home%%:40_SG_ETCFS:gpfs1:6:   type            = mmfs
%%home%%:40_SG_ETCFS:gpfs1:7:   account         = false
%%home%%:50_SG_MOUNT:gpfs1::rw:mtime:atime:::::::::::::::::::
%%home%%:60_SG_DISKS:gpfs1:1:gpfs4nsd:0:2:dataOnly:AC10149954467F0B:nsd:::other::hdisk:cmd::::ready::silver::::::
%%home%%:60_SG_DISKS:gpfs1:2:gpfs5nsd:0:1:dataAndMetadata:AC10149A544A5AEE:nsd:::other::hdisk:cmd::::ready::system:::
:::
%%home%%:60_SG_DISKS:gpfs1:3:gpfs1nsd:0:3:dataAndMetadata:AC10149954467F07:nsd:::other::hdisk:cmd::::::system::::::
%%home%%:60_SG_DISKS:gpfs1:4:gpfs2nsd:0:4:dataOnly:AC10149954467F09:nsd:::other::hdisk:cmd::::::silver::::::
%%home%%:60_SG_DISKS:gpfs1:5:gpfs3nsd:0:4:dataOnly:AC10149954467F0A:nsd:::other::hdisk:cmd::::::silver::::::
%%home%%:62_BK_LSDISK:gpfs1:1:mmlsdisk::HEADER:version:reserved:reserved:nsdName:driverType:sectorSize:failureGroup:m
etadata:data:status:availability:diskID:storagePool:remarks:numQuorumDisks:readQuorumValue:writeQuorumValue:diskSizeK
B:diskUID:
%%home%%:62_BK_LSDISK:gpfs1:2:mmlsdisk::0:1:::gpfs4nsd:nsd:512:2:no:yes:ready:up:1:silver:desc:3:2:2:10485760:AC1014
9A544AAAC1:
%%home%%:62_BK_LSDISK:gpfs1:3:mmlsdisk::0:1:::gpfs5nsd:nsd:512:1:yes:yes:ready:up:2:system:desc:3:2:2:10485760:AC1014
9A544AAABF:
```

%%home%%:62_BK_LSDISK:gpfs1:4:mmlsdisk::0:1:::gpfs1nsd:nsd:512:3:yes:yes:ready:up:3:system:desc:3:2:2:104857600:AC10149A544E25C5:
%%home%%:62_BK_LSDISK:gpfs1:5:mmlsdisk::0:1:::gpfs2nsd:nsd:512:4:no:yes:ready:up:4:silver::3:2:2:104857600:AC10149A544E25C1:
%%home%%:62_BK_LSDISK:gpfs1:6:mmlsdisk::0:1:::gpfs3nsd:nsd:512:4:no:yes:ready:up:5:silver::3:2:2:104857600:AC10149A544E25C3:
%%home%%:63_BK_LSFS:gpfs1:1:mmlsfs::0:1:::gpfs1:stripeMethod:roundRobin:
%%home%%:63_BK_LSFS:gpfs1:2:mmlsfs::0:1:::gpfs1:logicalSectorSize:512:
%%home%%:63_BK_LSFS:gpfs1:3:mmlsfs::0:1:::gpfs1:minFragmentSize:16384:
%%home%%:63_BK_LSFS:gpfs1:4:mmlsfs::0:1:::gpfs1:inodeSize:4096:
%%home%%:63_BK_LSFS:gpfs1:5:mmlsfs::0:1:::gpfs1:indirectBlockSize:16384:
%%home%%:63_BK_LSFS:gpfs1:6:mmlsfs::0:1:::gpfs1:defaultMetadataReplicas:1:
%%home%%:63_BK_LSFS:gpfs1:7:mmlsfs::0:1:::gpfs1:maxMetadataReplicas:2:
%%home%%:63_BK_LSFS:gpfs1:8:mmlsfs::0:1:::gpfs1:prefetchBuffers:0:
%%home%%:63_BK_LSFS:gpfs1:9:mmlsfs::0:1:::gpfs1:defaultDataReplicas:1:
%%home%%:63_BK_LSFS:gpfs1:10:mmlsfs::0:1:::gpfs1:maxDataReplicas:2:
%%home%%:63_BK_LSFS:gpfs1:11:mmlsfs::0:1:::gpfs1:blockAllocationType:cluster:
%%home%%:63_BK_LSFS:gpfs1:12:mmlsfs::0:1:::gpfs1:maxExpectedDiskIOLatency:0:
%%home%%:63_BK_LSFS:gpfs1:13:mmlsfs::0:1:::gpfs1:fileLockingSemantics:nfs4:
%%home%%:63_BK_LSFS:gpfs1:14:mmlsfs::0:1:::gpfs1:ACLSemantics:all:
%%home%%:63_BK_LSFS:gpfs1:15:mmlsfs::0:1:::gpfs1:estimatedAverageFilesize:1048576:
%%home%%:63_BK_LSFS:gpfs1:16:mmlsfs::0:1:::gpfs1:numNodes:32:
%%home%%:63_BK_LSFS:gpfs1:17:mmlsfs::0:1:::gpfs1:maxConcurrentIOOperationsPerDisk:1:
%%home%%:63_BK_LSFS:gpfs1:18:mmlsfs::0:1:::gpfs1:blockSize:524288:
%%home%%:63_BK_LSFS:gpfs1:19:mmlsfs::0:1:::gpfs1:quotasAccountingEnabled:none:
%%home%%:63_BK_LSFS:gpfs1:20:mmlsfs::0:1:::gpfs1:quotasEnforced:none:
%%home%%:63_BK_LSFS:gpfs1:21:mmlsfs::0:1:::gpfs1:defaultQuotasEnabled:none:
%%home%%:63_BK_LSFS:gpfs1:22:mmlsfs::0:1:::gpfs1:filesetdfEnabled:no:
%%home%%:63_BK_LSFS:gpfs1:23:mmlsfs::0:1:::gpfs1:filesystemVersion:14.10 (4.1.0.4):
%%home%%:63_BK_LSFS:gpfs1:24:mmlsfs::0:1:::gpfs1:filesystemVersionLocal:14.10 (4.1.0.4):
%%home%%:63_BK_LSFS:gpfs1:25:mmlsfs::0:1:::gpfs1:filesystemVersionManager:14.10 (4.1.0.4):
%%home%%:63_BK_LSFS:gpfs1:26:mmlsfs::0:1:::gpfs1:filesystemVersionOriginal:14.10 (4.1.0.4):
%%home%%:63_BK_LSFS:gpfs1:27:mmlsfs::0:1:::gpfs1:filesystemHighestSupported:14.10 (4.1.0.4):
%%home%%:63_BK_LSFS:gpfs1:28:mmlsfs::0:1:::gpfs1:create-time:Fri Oct 24 15%3A38%3A42 2014:
%%home%%:63_BK_LSFS:gpfs1:29:mmlsfs::0:1:::gpfs1:aggressivenessLevelOfTokensPrefetch:0:
%%home%%:63_BK_LSFS:gpfs1:30:mmlsfs::0:1:::gpfs1:supportForLargeLUNs:yes:
%%home%%:63_BK_LSFS:gpfs1:31:mmlsfs::0:1:::gpfs1:DMAPIEnabled:no:
%%home%%:63_BK_LSFS:gpfs1:32:mmlsfs::0:1:::gpfs1:logfileSize:419430:
%%home%%:63_BK_LSFS:gpfs1:33:mmlsfs::0:1:::gpfs1:exactMtime:yes:
%%home%%:63_BK_LSFS:gpfs1:34:mmlsfs::0:1:::gpfs1:suppressAtime:no:
%%home%%:63_BK_LSFS:gpfs1:35:mmlsfs::0:1:::gpfs1:strictReplication:whenpossible:
%%home%%:63_BK_LSFS:gpfs1:36:mmlsfs::0:1:::gpfs1:fastEAenabled:yes:
%%home%%:63_BK_LSFS:gpfs1:37:mmlsfs::0:1:::gpfs1:filesystemFeatures:FastEA; StripedLogs; 4KAligned:
%%home%%:63_BK_LSFS:gpfs1:38:mmlsfs::0:1:::gpfs1:encryption:no:
%%home%%:63_BK_LSFS:gpfs1:39:mmlsfs::0:1:::gpfs1:maxNumberOfInodes:112768:
%%home%%:63_BK_LSFS:gpfs1:40:mmlsfs::0:1:::gpfs1:afmFilesets:no:
%%home%%:63_BK_LSFS:gpfs1:41:mmlsfs::0:1:::gpfs1:maxSnapshotId:1:
%%home%%:63_BK_LSFS:gpfs1:42:mmlsfs::0:1:::gpfs1:UID:AC10149A%3A544AAAC2:
%%home%%:63_BK_LSFS:gpfs1:43:mmlsfs::0:1:::gpfs1:filesetCount:2:
%%home%%:63_BK_LSFS:gpfs1:44:mmlsfs::0:1:::gpfs1:logReplicas:0:
%%home%%:63_BK_LSFS:gpfs1:45:mmlsfs::0:1:::gpfs1:defaultBlockGroupFactor:1:
%%home%%:63_BK_LSFS:gpfs1:46:mmlsfs::0:1:::gpfs1:defaultWriteAffinityDepth:0:
%%home%%:63_BK_LSFS:gpfs1:47:mmlsfs::0:1:::gpfs1:defaultFailureGroupMaskBits:0:
%%home%%:63_BK_LSFS:gpfs1:48:mmlsfs::0:1:::gpfs1:is4KAligned:yes:
%%home%%:63_BK_LSFS:gpfs1:49:mmlsfs::0:1:::gpfs1:ccrCompatible:yes:

```
%%home%%:63_BK_LSFS:gpfs1:50:mmlsfs::0:1:::gpfs1:rapidRepairEnabled:yes:
%%home%%:63_BK_LSFS:gpfs1:51:mmlsfs::0:1:::gpfs1:write-cache-threshold:0:
%%home%%:63_BK_LSFS:gpfs1:52:mmlsfs::0:1:::gpfs1:storagePools:system;silver:
%%home%%:63_BK_LSFS:gpfs1:53:mmlsfs::0:1:::gpfs1:perfilesetQuotas:no:
%%home%%:63_BK_LSFS:gpfs1:54:mmlsfs::0:1:::gpfs1:mountPriority:0:
%%home%%:63_BK_LSFS:gpfs1:55:mmlsfs::0:1:::gpfs1:defaultMountPoint:%2Fgpfs1:
%%home%%:63_BK_LSFS:gpfs1:56:mmlsfs::0:1:::gpfs1:defaultDriveLetter::
%%home%%:63_BK_LSFS:gpfs1:57:mmlsfs::0:1:::gpfs1:automaticMountOption:yes:
%%home%%:63_BK_LSFS:gpfs1:58:mmlsfs::0:1:::gpfs1:additionalMountOptions:none:
%%home%%:63_BK_LSFS:gpfs1:59::::::::maxNumberOfInodesOfRoot:112768:112768:
%%home%%:64_BK_FILESET:gpfs1:1:mmlsfileset::HEADER:version:reserved:reserved:filesystemName:filesetName:id:rootInode:
status:path:parentId:created:inodes:dataInKB:comment:filesetMode:afmTarget:afmState:afmMode:afmFileLookupRefreshInter
val:afmFileOpenRefreshInterval:afmDirLookupRefreshInterval:afmDirOpenRefreshInterval:afmAsyncDelay:reserved:afmExpira
tionTimeout:afmRPO:afmLastPSnapId:inodeSpace:isInodeSpaceOwner:maxInodes:allocInodes:inodeSpaceMask:afmShowHomeSnapsh
ots:afmNumReadThreads:reserved:afmReadBufferSize:afmWriteBufferSize:afmReadSparseThreshold:afmParallelReadChunkSize:a
fmParallelReadThreshold:snapId:afmNumFlushThreads:afmPrefetchThreshold:afmAllowExpiration:permChangeFlag:afmParallelW
riteThreshold:freeInodes:reserved:afmParallelWriteChunkSize:afmNumWriteThreads:reserved:
%%home%%:64_BK_FILESET:gpfs1:2:mmlsfileset::0:1:::gpfs1:root:0:3:Linked:%2Fgpfs1:--:Fri Oct 24 15%3A38%3A40
2014:-:-:root
fileset:off:-:-:-:-:-:-:-:-:-:-:-:-:0:1:112768:112768:0:-:-:-:-:-:-:-:-:0:-:-:-:chmodAndSetacl:-:108714::-:-:-:
%%home%%:64_BK_FILESET:gpfs1:3:mmlsfileset::0:1:::gpfs1:tstfset:1:98688:Linked:%2Fgpfs1%2Ftstfset:0:Fri Oct 24
15%3A40%3A33 2014:-:-::off:-:-:-:-:-:-:-:-:-:-:-:-:-:0:0:0:0:0:-:-:-:-:-:-:-:-:0:-:-:-:chmodAndSetacl:-:0::-:-:-:
%%home%%:65_BK_POLICY:gpfs1:1:No policy file was installed for file system 'gpfs1'.
%%home%%:66_BK_QUOTA:gpfs1:1:gpfs1: no quota management enabled
%%home%%:66_BK_QUOTA:gpfs1:2:gpfs1: no quota management enabled
%%home%%:66_BK_QUOTA:gpfs1:3:gpfs1: no quota management enabled
%%home%%:69_BK_LSPOOL:gpfs1:1:mmlspool::HEADER:version:reserved:reserved:filesystemName:poolName:poolId:blockSize:usa
ge:maxDiskSize:layoutMap:allowWriteAffinity:writeAffinityDepth:blockGroupFactor:
%%home%%:69_BK_LSPOOL:gpfs1:2:mmlspool::0:1:::gpfs1:system:0:524288:dataAndMetadata:202984914944:cluster:no:0:1:
%%home%%:69_BK_LSPOOL:gpfs1:3:mmlspool::0:1:::gpfs1:silver:65537:524288:dataOnly:905995878400:cluster:no:0:1:
```

Generating the scripts for creating the original file system configuration

The following script named **createfsfiles.sh** is provided “as-is”, and has been tested in our lab environment. The script performs the following operations:

- Calls the **mmbackupconfig** command to generate the file system configuration data.
- Calls the **mmrestoreconfig** with the **-F <fsname>_queryconfig** file to generate the file containing the build information.
- Filters the `queryconfig` file to generate the **mmcrfs** script and the stanza file for NSD creation.

Run the script from the directory installed:

```
./createfsfiles.sh <fsname>
```

The script generates the following files in the current directory:

- `<fsname>_backupconfig`: The output of the **mmbackupconfig** command.
- `mmcrfs_<fsname>`: The script for creating the Spectrum Scale file system, which calls the **mmcrfs** command with custom parameters from the original configuration (block size, version information, inode size, and other file system-specific parameters).

- <fsname>_stanza: The stanza file for the **mmcrfs** command in the previous script.
- <fsname>_queryconfig: A file containing the build information for the Spectrum Scale file system and NSDs.

The output of the **createfsfiles.sh** script is shown in Example A-3.

Example A-3 createfsfiles.sh script

```
#====================================================================
#   createfsfiles ksh
# ===================================================================
#!/bin/ksh

if [ $# -ne 1 ]
then
        echo Usage: $0 fs
        echo
        exit 1
fi

fs=$1
fs_stanzas="$fs"_stanzas
rm $fs.backupconfig $fs.queryconfig
echo "========================================="
echo "createfsfiles: Creating filesystem \"$fs\" datafiles for image restore"
echo "========================================="
echo "mmbackupconfig $fs -o $fs.backupconfig"
        mmbackupconfig $fs -o $fs.backupconfig
        if [ $? -ne 0 ]
        then
                echo "mmbackupconfig returned non-zero rc=$?"
                exit 1
        fi
        echo "mmbackupconfig returned $?"
echo "mmrestoreconfig $fs -i $fs.backupconfig -F $fs.queryconfig"
        mmrestoreconfig $fs -i $fs.backupconfig -F $fs.queryconfig
        if [ $? -ne 0 ]
        then
                echo "mmrestoreconfig returned non-zero rc=$?"
                exit 1
        fi
        echo "mmrestoreconfig returned $?"
echo "========================================="
        cat $fs.queryconfig | sed -e "s/^# mmcrfs/mmcrfs/g;s/FS_NAME/$fs/g;s/NSD_DISKS/-F $fs_stanzas/g" > mmcrfs_$fs
        cat $fs.queryconfig | sed -e "s/^#/
/g;s/mmcrfs/#mmcrfs/g;s/When/#When/g;s/enforcement/#enforcement/g;s/If/#If/g" > $fs_stanzas
        echo "chmod 777 $fs_stanzas mmcrfs_$fs"
        chmod 777 $fs_stanzas mmcrfs_$fs
```

Restoring the file system configuration

The following scenario presents a backup and restore example based on the **createfsfiles.sh** script, for a Spectrum Scale file system named gpfs1.

The following steps are performed:

1. Run the **createfsfiles.sh** script with **gpfs1** as the file system parameter. See Example A-4.

Example A-4 Running createfsfiles.sh script

```
root@fsaix4:/work>./createfsfiles.sh gpfs1
rm: gpfs1.backupconfig: No such file or directory
rm: gpfs1.queryconfig: No such file or directory
=========================================
createfsfiles: Creating filesystem "gpfs1" datafiles for image restore
=========================================
mmbackupconfig gpfs1 -o gpfs1.backupconfig

mmbackupconfig: Processing file system gpfs1 ...
mmbackupconfig: Command successfully completed
mmbackupconfig returned 0
mmrestoreconfig gpfs1 -i gpfs1.backupconfig -F gpfs1.queryconfig
mmrestoreconfig: Configuration file successfully created in gpfs1.queryconfig
mmrestoreconfig: Command successfully completed
mmrestoreconfig returned 0
=========================================
chmod 777 gpfs1_stanzas mmcrfs_gpfs1
```

The result of running the script are the files shown in Example A-5.

Example A-5 Generated files of createfsfiles.sh script

```
root@fsaix4:/work>ls -al
total 64
drwxr-xr-x    2 root     system          256 Oct 28 21:27 .
drwxr-xr-x   25 root     system         4096 Oct 28 21:29 ..
-rwxr--r--    1 root     system         1479 Oct 27 17:28 createfsfiles.sh
-rw-r--r--    1 root     system         9295 Oct 28 21:27 gpfs1.backupconfig
-rw-r--r--    1 root     system         2383 Oct 28 21:27 gpfs1.queryconfig
-rwxrwxrwx    1 root     system         2387 Oct 28 21:27 gpfs1_stanzas
-rwxrwxrwx    1 root     system         2386 Oct 28 21:27 mmcrfs_gpfs1
```

2. We delete the file system to re-create it from scratch as shown in Example A-6.

Example A-6 Unmount and delete the file system gpfs1

```
root@fsaix4:/work>mmunmount gpfs1 -a
Tue Oct 28 21:29:21 EDT 2014: mmunmount: Unmounting file systems ...
root@fsaix4:/work>mmdelfs gpfs1
All data on the following disks of gpfs1 will be destroyed:
    gpfs4nsd
    gpfs5nsd
    gpfs1nsd
    gpfs2nsd
```

```
    gpfs3nsd
Completed deletion of file system /dev/gpfs1.
mmdelfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

We also delete the NSDs from the Spectrum Scale cluster configuration as shown in Example A-7.

Example A-7 Deleting the NSD configuration

```
root@fsaix4:/work> mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd;gpfs5nsd"
mmdelnsd: Processing disk gpfs1nsd
mmdelnsd: Processing disk gpfs2nsd
mmdelnsd: Processing disk gpfs3nsd
mmdelnsd: Processing disk gpfs4nsd
mmdelnsd: Processing disk gpfs5nsd
mmdelnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

3. In this step, re-create the original file system. Before creating the file system, create the NSD configuration. The stanza file provided in step 1 needs to be populated with the real device names according to your environment. For our case, the stanza file populated with the real device names is shown in Example A-8.

Example A-8 Populating the stanza file with real device names for creating the NSDs

```
%nsd: device=hdisk10
    usage=dataOnly
    failureGroup=2
    pool=silver

  %nsd: device=hdisk7
    usage=dataAndMetadata
    failureGroup=1
    pool=system

  %nsd: device=hdisk11
    usage=dataAndMetadata
    failureGroup=3
    pool=system

  %nsd: device=hdisk12
    usage=dataOnly
    failureGroup=4
    pool=silver

  %nsd: device=hdisk13
    usage=dataOnly
    failureGroup=4
    pool=silver
```

Tip: Use the information provided in the stanza file to identify the device sizes under the section “Disk Information”. See Example A-9 on page 522.

Example A-9 Disk Information for NSD rebuild

```
###### Disk Information #######
 # Number of disks 5
 # gpfs4nsd 104857600
 # gpfs5nsd 10485760
 # gpfs1nsd 104857600
 # gpfs2nsd 104857600
 # gpfs3nsd 104857600

 # Number of Storage Pools 2
 # silver 314572800
 # system 115343360
```

Based on the information in Example A-9, we choose the new hdisk devices to match the original sizes of the NSDs and storage pools:

- There were five disks: 4 x 100 GB and 1 x 10 GB.
- The capacity of the system pool was 115343360 KB, so we need to provide 100 GB + 10 GB logical unit numbers (LUNs).
- The capacity of the silver pool was 314572800 KB, so we need to provide 3 x 100 GB LUNs.

We run the **mmcrnsd** command with the stanza file populated with the real hdisk devices. See Example A-10.

Example A-10 Creating the NSD devices

```
root@fsaix4:/work> mmcrnsd -F ./gpfs1_stanzas
mmcrnsd: Processing disk hdisk10
mmcrnsd: Processing disk hdisk7
mmcrnsd: Processing disk hdisk11
mmcrnsd: Processing disk hdisk12
mmcrnsd: Processing disk hdisk13
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Run the **mmcrfs_gpfs1** script from the current directory containing all processing files. See Example A-11.

Example A-11 Creating the Spectrum Scale file system with the original attributes

```
root@fsaix4:/work>./mmcrfs_gpfs1

The following disks of gpfs1 will be formatted on node tsmcl1:
    gpfs6nsd: size 102400 MB
    gpfs7nsd: size 10240 MB
    gpfs8nsd: size 102400 MB
    gpfs9nsd: size 102400 MB
    gpfs10nsd: size 102400 MB
Formatting file system ...
Disks up to size 1.0 TB can be added to storage pool system.
Disks up to size 1.7 TB can be added to storage pool silver.
Creating Inode File
Creating Allocation Maps
Creating Log Files
```

```
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Formatting Allocation Map for storage pool silver
Completed creation of file system /dev/gpfs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

4. Restore the file system configuration by using the **mmrestoreconfig** command. See Example A-12.

Example A-12 Restoring the extended attributes for file system gpfs1

```
root@fsaix4:/work> mmrestoreconfig gpfs1 -i ./gpfs1.backupconfig
------------------------------------------------------
Configuration restore of gpfs1 begins at Tue Oct 28 23:06:21 EDT 2014.
------------------------------------------------------
mmrestoreconfig: Checking disk settings for gpfs1:
mmrestoreconfig: Checking the number of storage pools defined for gpfs1.
mmrestoreconfig: Checking storage pool names defined for gpfs1.
mmrestoreconfig: Checking storage pool size for 'silver'.
mmrestoreconfig: Checking storage pool size for 'system'.

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:

mmrestoreconfig: Checking fileset configurations for gpfs1:
Fileset tstfset created with id 1 root inode 66816.

mmrestoreconfig: Mounting file system gpfs1 to link filesets.
Fileset tstfset linked at /gpfs1/tstfset
mmrestoreconfig: Fileset link status has been restored.  Unmounting file system
gpfs1.
Tue Oct 28 23:07:21 EDT 2014: mmumount: Unmounting file systems ...

mmrestoreconfig: Checking policy rule configuration for gpfs1:
mmrestoreconfig: No policy rules installed in backed up filesystem gpfs1.

mmrestoreconfig: Checking filesystem attribute configuration for gpfs1:
mmrestoreconfig: Command successfully completed
```

5. Mount the file system `gpfs1` on all cluster nodes as shown in Example A-13.

Example A-13 Mounting the Spectrum Scale file system on all cluster nodes

```
root@fsaix4:/work>mmmount gpfs1 -a
Tue Oct 28 23:32:46 EDT 2014: mmmount: Mounting file systems ...

root@fsaix4:/work>mmdf gpfs1
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 965 GB)
gpfs7nsd             10485760        1 yes      yes        10104320 ( 96%)          1360 ( 0%)
gpfs8nsd            104857600        3 yes      yes       104475136 (100%)          1136 ( 0%)
                -------------                         -------------------- -------------------
(pool total)        115343360                            114579456 ( 99%)          2496 ( 0%)
```

```
Disks in storage pool: silver (Maximum disk size allowed is 1.5 TB)
gpfs6nsd            104857600        2 no       yes       104791040 (100%)           880 ( 0%)
gpfs9nsd            104857600        4 no       yes       104791040 (100%)           880 ( 0%)
gpfs10nsd           104857600        4 no       yes       104791040 (100%)           880 ( 0%)
                -------------                             -------------------- -------------------
(pool total)        314572800                              314373120 (100%)          2640 ( 0%)

                =============                             ==================== ===================
(data)              429916160                              428952576 (100%)          5136 ( 0%)
(metadata)          115343360                              114579456 ( 99%)          2496 ( 0%)
                =============                             ==================== ===================
(total)             429916160                              428952576 (100%)          5136 ( 0%)

Inode Information
-----------------
Number of used inodes:            4039
Number of free inodes:          108857
Number of allocated inodes:     112896
Maximum number of inodes:       112896
```

B

How to obtain an IBM Spectrum Scale trial version

Note: The information here is offered as-is. It is up to date at the moment of writing the publication. Refer to the IBM source on the IBM Spectrum Scale FAQ for additional information:

http://ibm.co/1EwJtZz

Prospective customers of IBM Spectrum Scale can avail the option of a free 90 day trial program.

Please contact your IBM sales representative to apply for this trial evaluation.

ISVs should obtain the software and licenses via the IBM Partner World Software Access catalog at:
https://www-304.ibm.com/jct01004c/partnerworld/partnertools/eorderweb/ordersw.do

Please contact scale@us.ibm.com for information regarding the IBM Spectrum Scale trial program.

© Copyright IBM Corp. 2015. All rights reserved.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM System x Reference Architecture for Hadoop: IBM InfoSphere BigInsights Reference Architecture*, REDP-5009
- *IBM PowerKVM Configuration and Use*, SG24-8231
- *IBM PowerVM Virtualization Managing and Monitoring*, SG24-7590
- *IBM PowerVM Virtualization Introduction and Configuration*, SG24-7940
- *Exploiting IBM AIX Workload Partitions*, SG24-7955
- *IBM Linear Tape File System Enterprise Edition V1.1.1.2: Installation and Configuration Guide*, SG24-8143
- *IBM Tivoli Storage Manager as a Data Protection Solution*, SG24-8134

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *IBM Spectrum Scale v4.1: Concepts, Planning, and Installation Guide*, GA76-0441-01
- *IBM Spectrum Scale v4.1: Data Management API Guide*, GA76-0442-01
- *IBM Spectrum Scale v4.1: Advanced Administration Guide*, SC23-7032-01
- *IBM Security Key Lifecycle Manager: Installation and Configuration Guide*, SC27-5335-01

Online resources

These websites are also relevant as further information sources:

- IBM Watson

  http://www.ibm.com/smarterplanet/us/en/ibmwatson
- Deploying a big data solution using IBM GPFS-FPO

  http://public.dhe.ibm.com/common/ssi/ecm/en/dcw03051usen/DCW03051USEN.PDF

© Copyright IBM Corp. 2015. All rights reserved.

- IBM Spectrum Scale Frequently Asked Questions (FAQ)

  http://ibm.co/1IKO6PN

Help from IBM

IBM Support and downloads

ibm.com/support

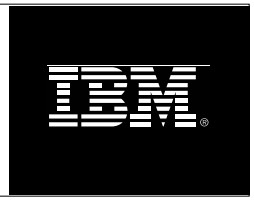
IBM Global Services

ibm.com/services

IBM Spectrum Scale (formerly GPFS)

SG24-8254-00

ISBN 0738440736



(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

SG24-8254-00

ISBN 0738440736

Printed in U.S.A.

Get connected